KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ TINKLŲ KATEDRA

Gintarė Grigonytė

Priklausomybių gramatikos taikymas lietuvių kalbos apdorojime

Magistro darbas

Darbo vadovas
dr. G. Raškinis

Kaunas, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ TINKLŲ KATEDRA

Gintarė Grigonytė

Priklausomybių gramatikos taikymas lietuvių kalbos apdorojime

Magistro darbas

Kalbos konsultantė
Lietuvių k. katedros lekt.
dr. J. Mikelionienė
2006-05-25

Vadovas
dr. G. Raškinis
2006-05-25

Recenzentas
doc. dr. E. Karčiauskas
2006-05-26

Atliko
IFM-0/2 gr. stud.
Gintarė Grigonytė
2006-05-25

Kaunas, 2006

SUMMARY

Dependency Grammar in Lithuanian Language Processing

Lithuanian language is quite in an early stage of language processing. And therefore has a high demand on automated tools like taggers, parsers, word sense disambiguators etc. During the last 10 years only a few researchers were attempting to create a parser for Lithuanian language. However none of them are used in practices nowadays.

The process of designing and implementing rule based parser for Lithuanian language is presented in this paper. Rules and constraints of the formal grammar follow the principles of *Dependency Grammar*. Necessary language recourses were made up at the Computational Centre of Linguistics.

Research area analysis and overview of the most popular methodologies is presented here as well.

Syntax parser of the Lithuanian language was evaluated against the Gold Standard and gave 80,2 % of accuracy of in recognizing parts of the sentence

Key words: parser, syntax, dependency grammar, rule based.

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

1. ĮVADAS

Programų sistemų inžinerija – dinamiška ir besiplečianti mokslo šaka, kuri gali būti sėkmingai pritaikoma tarpdisciplininių sričių aktualioms problemoms spręsti. Viena iš tokių sričių yra kompiuterinė lingvistika.

Kompiuterinė lingvistika apima daugelį siauresnių krypčių, kurių tolimesniam vystymuisi reikalinga automatinė sintaksinė kalbos analizė. Sintaksinė analizė yra svarbus etapas beveik visiems kalbos technologijos uždaviniams atlikti.

Projekte įgyvendinamo sintaksinio analizatoriaus, paremto priklausomybių gramatika, pritaikomumo sričių yra daug: gramatikos tikrintuvų kūrimas, informacijos iš elektroninių tekstų išgavimas ir ištraukimas, sudėtingesnių paieškos sistemų kūrimas, automatinės vertimo sistemos iš lietuvių kalbos į kitas kalbas ir atvirkščiai, lietuvių kalbos mokymas ir kiti uždaviniai, kuriems reikalinga kalbos analizė.

Projekto **teorinė ir praktinė svarba** yra ta, kad jis vienija programų sistemų inžinerijos metodus ir automatinei sintaksinei analizei įgyvendinti pasirinktą kelią.

1.1. Tikslai ir uždaviniai

Šio projekto **tikslas** yra rasti tinkamą sprendimą lietuvių kalbos sintaksinei analizei atlikti ir, remiantis programų sistemos inžinerijos metodologija, jį realizuoti.

Apibrėžtam tikslui pasiekti, keliami šie **uždaviniai:**

1. Išanalizuoti ir įsisavinti sintaksinės analizės metodikas, ištirti bei apžvelgti alternatyvius projektus.
2. Išanalizuoti kuriamai sistemai keliamus reikalavimus ir parinkti tinkamiausią sprendimo kelią.
3. Suprojektuoti lietuvių kalbos automatinės sintaksinės analizės sistemą.
4. Sukurti lietuvių kalbos automatinės sintaksinės analizės sistemą.
5. Ištirti sukurtą sistemą, įvertinti jos kokybę ir palyginti su išanalizuotomis sintaksinės analizės alternatyviomis sistemomis.

1.2. Dokumento paskirtis

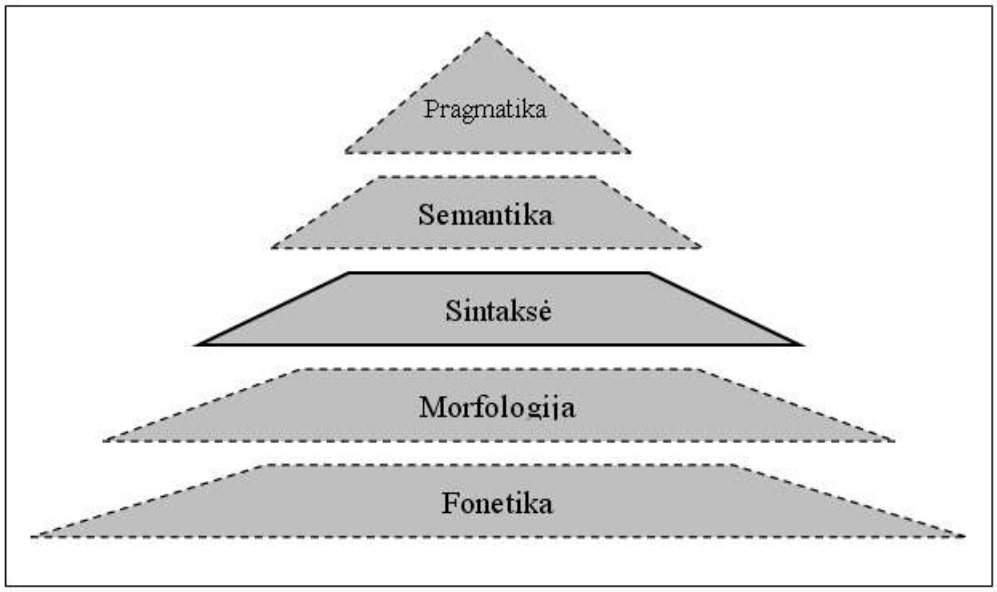
Dokumente pateikiamos aktualios sintaksinės analizės problemos, metodai toms problemoms spręsti. Supažindinama su svarbiausiomis sintaksinės analizės strategijomis, formaliąja gramatika, pateikiama projektuojamos sistemos ir jos sprendimų būdų analizė, pristatomas ir pagrindžiamas pasirinktas inžinerinis problemos sprendimo būdas. Dokumente pateikiamas pasirinktas ir realizuotas metodas lietuvių kalbos sintaksinei analizei atlikti. Taip pat įvertinama realizuota programinės įrangos kokybė, nurodomos gairės tolimesniam programų sistemos plėtrai.

2. ANALITINĖ DALIS

Natūralios kalbos automatizavimas susideda iš penkių kalbos nagrinėjimo lygių: fonetikos, morfologijos, sintaksės, semantikos ir pragmatikos (žr. pav.1). Pilnas kalbos automatizavimas yra nuoseklus visų lygių įgyvendinimo rezultatas [21].

Šio darbo **objektas** yra lietuvių kalbos sintaksės lygmens automatizavimas.

Sintaksės lygmens automatizavimas yra **aktualus** lietuvių kalbos automatizavimo uždavinys, kadangi be sintaksinės analizės negalima atlikti semantinės ir pragmatinės analizės [13] (teorinė svarba), kurti mašininio vertimo sistemų, gramatikos tikrintuvų, automatinių informacijos apdorojimo įrankių (praktinė svarba).



Pav. 1. Natūralios kalbos nagrinėjimo lygiai.

Kitas svarbus projekto aspektas yra **naujumas**: projekte ne tik teoriškai išanalizuotas, bet ir realizuotas pasirinktas automatinės sintaksinės analizės, paremtos priklausomybių gramatika, kelias.

Šios dokumento dalies skyriuose pristatoma sintaksinė analizė ir priklausomybių gramatikos taikymo principai. Pateikiama literatūros, susijusios su priklausomybių gramatikos tyrimais ir taikymais, analizė, kuria siekiama apžvelgti projekto taikymo sritį, kitų pasaulio kalbų lingvistikos pasiekimus automatinės sintaksinės analizės srityje, pasiruošti specifikuoti ir projektuoti sistemos reikalavimus.

2.1. Sintaksinė analizė

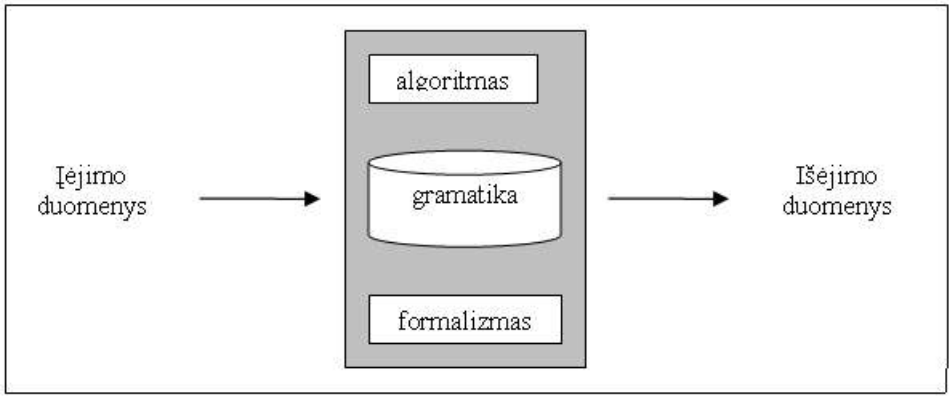
Sintaksinė analizė yra vienas geriausiai išanalizuotų ir suprantamų informatikos mokslo uždavinių. Sintaksinė analizė sėkmingai taikoma daugelyje sričių:

- Informatikoje – kuriant kompiliatorius, DB sąsajoms, dirbtinio intelekto uždaviniams,
- Lingvistikoje – tekstų analizei, tekstynų analizei, mašininiam vertimui,
- Paruošiant ir transformuojant dokumentus,
- Nustatant ryšius cheminėse formulėse, atpažįstant chromosomas ir kt. [14].

Dokumente pristatoma, kaip sintaksinė analizė suprantama ir taikoma kompiuterinėje lingvistikoje. Sintaksinė analizė (angl. *parsing*) – tai struktūrinis simbolių eilutės (žodžio, žodžių junginio, sakinio) aprašas, paremtas tam tikra gramatika. Ši uždavinį sprendžianti programų sistema vadinama sintaksiniu analizatoriumi (angl. *parser*) [17].

Sintaksiniuose analizatoriuose gali būti naudojami įvairūs gramatikos specifikavimo formatai: produkcijų sistemos, būsenų perėjimo tinklai, vakansijos ir užpildai, analizės medis gali būti konstruojamas iš viršaus į apačią, iš apačios į viršų, atliekant paiešką iš pradžių į gylį arba iš pradžių į plotį [7].

Kuriant sintaksinį analizatorių yra reikalingi šie elementai: formalizmas gramatikai aprašyti; gramatika; algoritmas, sugebantis patikrinti, ar įėjimo duomenys tenkina gramatiką, ir jei taip, tai kokią struktūrą jis atitinka (žr. pav.2.).



Pav. 2. Sintaksinio analizatoriaus būtini komponentai.

Gramatikos formalizmas turi būti suprantamas, gramatika – adekvati, o algoritmas – efektyvus. Sintaksinė struktūra paprastai vaizduojama medžio pavidalo diagramomis.

2.2. Formalioji gramatika

Chomsky 1965 m. pateiktu formalios gramatikos aprašu remiasi beveik visų formaliųjų gramatikų tyrėjai ir sudarytojai, sintaksinių analizatorių ir kompiliatorių kūrėjai [7].

Kaip matyti iš 2.1. skyrelio, formalioji gramatika yra būtinas sintaksinio analizatoriaus komponentas. Formaliosios gramatikos yra matematikos mokslo šaka, todėl toliau pateikiami aprašai turės standartinį pavidalą, priimtą matematikoje.

Apibrėžimas: generatyvinė gramatika yra keturių komponentų (VN, VT, R, S) rinkinys, kai
(1) VN ir VT yra baigtinis simbolių rinkinys,
(2) $VN \cap VT = \varnothing$,
(3) R yra porų (P,Q) rinkinys, tokių kad:
(3a) $P \in (VN \cup VT)$
(3b) $Q \in (VN \cup VT)^*$,
(4) $S \in VN$.

Ši formali gramatika susideda iš keturių dalių: neterminalinių simbolių, terminalinių simbolių, taisyklių ir pradžios simbolio. Neterminalinių simbolių rinkinys yra VN, terminalinių simbolių rinkinys yra VT, R – taisyklių rinkinys, S – pradžios simbolis [21], [11].

Apsibrėžkime tokią gramatiką, kad:
VN = {`Vardas`, `Sakinys`, `Sąrašas`, `Pabaiga`}
VT = {`Jonas`, `Petras`, `Ona`, `,`, `ir`}
(pastaba: „`,`“ yra terminalinis simbolis).

VN ir VT (2) sankirta turi būti tuščia aibė, tai reiškia, kad terminalinių simbolių rinkinyje neturi būti nei vieno simbolio, esančio neterminalinių simbolių rinkinyje, ir atvirkščiai.

R yra taisyklių rinkinys (3), atitinkamai P ir Q yra kairiosios ir dešiniosios taisyklių pusės. Kiekvienas P turi susidėti iš vieno ar daugiau neterminalinių simbolių, ir kiekvienas Q turi būti sudarytas iš jokio arba vieno arba daugiau neterminalinių simbolių. Taisyklių rinkinys turi atitikti visus aukščiau išvardintus reikalavimus. Gramatiką papildome:

R = {(`Vardas`, `Jonas`), (`Vardas`, `Petras`), (`Vardas`, `Ona`), (`Sakinys`, `Vardas`), (`Sakinys`, `Sąrašas Pabaiga`), (`Sąrašas`, `Vardas`), (`Sąrašas`, `Sąrašas , Vardas`), (`,Vardas Pabaiga`, `ir Vardas`)}

Pradžios simbolis S turi būti neterminalinis simbolis, t. y. priklausyti VN:

S = `Sentence`

2.3. Sintaksinės analizės metodai

Šiame skyrelyje aptariamos galimos sintaksinės analizės strategijos ir pagrindžiamas projekte naudojamos strategijos pasirinkimas.

R. Hausser (2001) skiria dvi kalbos automatizavimo įrankių kūrimo strategijas: *greitą* (angl. *smart*) ir *vientisą* (angl. *solid*). Remiantis šiuos skirstymu, galima analizuoti ir sintaksinius analizatorius.

Sintaksiniai analizatoriai sukurti pagal *greitą* strategiją yra FSA analizatoriai ir statistiniai analizatoriai [19].

- Privalumai: šie analizatoriai yra nesudėtingi ir pigūs kalbos resursų prasme, nesigilina į probleminių sintaksės ryšių sprendimą teorine prasme.
- Trūkumai: neišbaigtumas, nepilnas duomenų padengimas, ribotas veikimo tikslumas.

Greitos strategijos sprendimai atrodo pigesni ir greičiau įgyvendinami, tačiau realybėje jų palaikymas, pakartotinis panaudojamumas ir optimizavimas yra brangus. Be to, veikimo tikslumo (angl. *accuracy*) didinimas yra neefektyvus ir sunkiai praktiškai įgyvendinamas.

Sintaksiniai analizatoriai, sukurti pagal *vientisą* strategiją, yra gramatikos taisyklių taikymu paremti sintaksiniai analizatoriai [21].

- Privalumai: *vientisos* strategijos analizatorių komponentai yra nuo pritaikymo nepriklausoma ir ilgalaikė investicija. Dėl sisteminės teorinės struktūros, sistemos komponentai gali būti lengvai prižiūrimi, pakartotinai panaudojami kitose sistemose, nesunkiai išplečiami ir tobulinami. Galima pasiekti didesnį tikslumą, geriau padengiami duomenys.
- Trūkumai: didelės laiko sąnaudos sprendimui realizuoti.

Lyginant abi sprendimų strategijas (*greitą* ir *vientisą*) svarbus faktorius yra kuriamos sistemos veikimo tikslumas. Pavyzdžiui, sistemai pasiekus 70 % veikimo tikslumą, jį dar galima tobulinti *vientisos* strategijos sprendimo atveju, kadangi rezultatai daugiausia priklauso nuo analizuojamą kalbą aprašančių taisyklių rinkinio, t. y. naujų taisyklių įvedimas gali pagerinti sistemos veikimo tikslumą iki

71 %. O *greitos* strategijos sprendimo atveju pasiekti 71 % tikslumą po to, kai sistema veikė 70 % tikslumu, yra labai sunku arba neįmanoma [21].

Projekte įgyvendintas sintaksinės analizės sprendimas atitinka *vientisos* strategijos būdą. Pagrindiniai kriterijai, nulėmę šį pasirinkimą, yra:

- Sistemos veikimo tikslumo gerinimo galimybė;
- Pakartotinis panaudojamumas;
- Išplečiamumas;
- Sistemos palaikymas.

Toliau apžvelgiamos *vientisos* strategijos populiariausi sprendimo būdai – pasaulio kalboms taikytos sintaksinės analizės metodikos: frazinė sintaksinė analizė ir priklausomybių gramatika paremta sintaksinė analizė.

Tarpusavyje minėtos metodikos skiriasi sintaksės struktūrų aprašymo tipais (sudedamųjų dalių struktūra ir priklausomybių struktūra) ir išvedimo duomenų formavimo būdais: iš viršaus į apačią (angl. *top-down*) ir iš apačios į viršų (angl. *bottom-up*) [6].

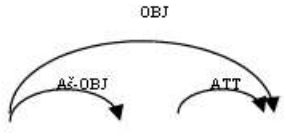
Analizė *iš viršaus į apačią* pradedama nuo stambiausio analizės elemento, paprastai tai būna sakinys, ir laipsniškai smulkinama iki frazių, toliau iki žodžių lygmens.

Analizė *iš apačios į viršų* paremta įėjimo duomenų analizavimu pradedant žodžiais, vėliau iš tų žodžių suformuotais junginiais, taip palaipsniui pereinama prie sakinio lygmens.

2.3.1. Priklausomybių analizė

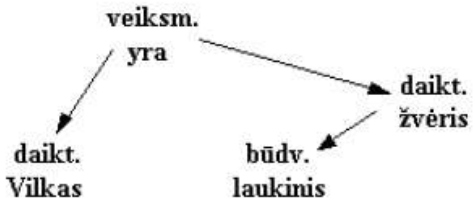
Pirmasis priklausomybių gramatikos teoriją išplėtojo prancūzų lingvistas Lucien Tesnière (*Éléments de syntaxe structurale*, 1959) [5]. Priklausomybių gramatika yra naudojama kalbų, turinčių negriežtą žodžių tvarką sakinyje, analizei. Tokios kalbos yra lenkų, čekų, rusų.

Mel'čuk 1987 m. pasiūlytos sintaksinės analizės esmė – priklausomybė (angl. *dependency*), jungianti du žodžius – valdantį ir valdomą. Kiekvieną priklausomybę apibūdina gramatinė funkcija (pavyzdžiui, subjektas ar objektas), nurodanti valdantįjį žodį [23]. Pavyzdžiui, sakinys *„Pasakykite man skrydžių laikus."* gali būti išanalizuotas kaip parodyta (5).



(5) Pasakykite man skrydžių laikus.

Priklausomybių medyje mazgai yra elementarūs įėjimo eilutės segmentai, o ryšiai nusako sintaksinius ryšius tarp elementaraus segmento ir nuo jo priklausomo priklausomybių struktūros pomedžio [17]. Dar vienas priklausomybių medžio pavyzdys sakiniui *„Vilkas yra laukinis žvėris.“* pateikiamas pav. 3.



Pav. 3. Priklausomybių struktūra

Priklausomybių analizė yra populiarus sintaksinės analizės būdas, daugiausiai taikytinas ne fleksinėms kalboms [11]. Šiuo metodu paremti toliau išvardinti žymiausi Europos kalbų sintaksiniai analizatoriai.

„Link Grammar“ sistemoje (Grinberg, Lafferty ir Sleator, 1995) kiekvienas žodyno žodis yra susiejamas su rinkiniu visų galimų funkcinių priklausomybių, nurodant jų kryptis. Sintaksiškai analizuojant, visos galimos sakinio žodžio priklausomybės yra peržiūrimos ir nustatomas kiekvienos jų tikimas/netikimas sakiniui [21].

„Functional Dependency Grammar“ sistemoje (Tapanainen ir Järvinen, 1997) pirmas sintaksinio analizatoriaus žingsnis yra pažymėti kiekvieną sakinio žodį visais galimais funkcijų tipais, toliau pritaikomas lingvistų sudarytas taisyklių rinkinys. Kiekvienos taisyklės pritaikymo metu, nustatomi valdantis ir valdomas žodžiai, taip pat gali būti panaikinami ankstesnės taisyklės sukurti, tačiau prieštaraujantys vėlesnei taisyklei ryšiai [17].

Priklausomybių analizės metodas buvo pritaikytas ir lietuvių kalbos sintaksinėje analizėje. Daudaravičius (2002) sakinį analizuoja kaip grafą, siejantį visus žodžius visais galimais būdais. Nereikalingi ryšiai atmetami projektiškumo kriterijaus pagalba.

Įgyvendintos sistemos ir galimi sprendimo būdai tarpusavyje skiriasi, tačiau turi šiuos bendrus bruožus: veikimo principą ir analizavimo apribojimus.

Priklausomybių gramatika paremto sintaksinio analizatoriaus veikimo **principas** yra toks: žodžių eilutei (nebūtinai sakiniui) reikia priskirti tinkamą priklausomybių rinkinį. Tai daroma atsižvelgiant į keletą apribojimų:

• **Vienumas** (angl. *unity*): išėjimo duomenys susiejami į vieną medį (6), t. y. visi sakinio žodžiai turi būti įtraukti į analizės medį, turintį vieną šaknį.

(6) Jonas stovi nuošaliau.

• **Unikalumas** (angl. *uniqueness*): kiekvienas žodis turi tik vieną viršūnę; tai reiškia, kad priklausomybių ryšiai formuoja medį, o ne grafą (7). Unikalumo kriterijų turi dauguma priklausomybių gramatikų, tačiau, pavyzdžiui, Hudsonas [11] siūlo žodžiui leisti turėti daugiau nei vieną viršūnę, t.y. tas pats žodis gali jungti keletą žodžių sakinyje (8).

(7) Jonukas ir Marytė stovi nuošaliau.

(8) Jonukas ir Marytė stovi nuošaliau.

Projekte remsimės požiūriu, kad vienas žodis gali turėti vieną ar daugiau viršūnių. Toks požiūris yra parankesnis lietuvių kalbai, kadangi sakinyje žodžiai paprastai yra susiję su vienu ar keliais kitais žodžiais [26].

• **Projektiškumas** (angl. *projectivity*): Jei žodis A yra valdomas žodžio B, tai visi žodžiai įsiterpiantys tarp A ir B yra tai pat priklausomi nuo B [25]. Kitais žodžiais tariant, šis reikalavimas reiškia, kad priklausomybių medyje negali būti tarpusavyje besikertančių ryšių. Sakiniai gali būti neprojektiški (9) ir projektiški (10).

(9) Vėliau dvigubai jis turėjo sumokėti.

(10) Vėliau jis turėjo sumokėti dvigubai.

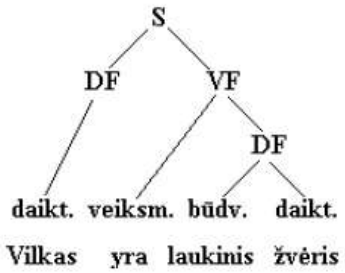
Kai kurios priklausomybių gramatikos, aprašančios kalbas, turinčias laisvą žodžių tvarką, leidžia neatsižvelgti į šį kriterijų. Lietuvių kalba taip pat turi laisvą žodžių tvarką [26]. Tačiau kalboje gausu junginių, kurie turi nusistovėjusią tvarką ir keisdami vietomis žodžius tokiuose

junginiuose gautume nevartojamas kalbos konstrukcijas [1]. Todėl projektiškumo kriterijus aktualus lietuvių kalbai.

• **Godumas** (angl. *eagerness*): Neprieštaraujančios taisyklėms ir apribojimams priklausomybės sakinyje nustatomos kaip įmanoma anksčiau [21].

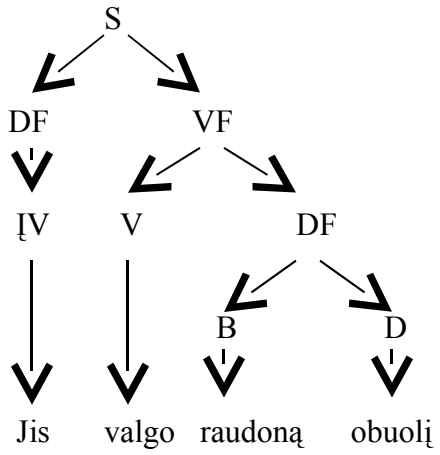
2.3.2. Frazių analizė

Frazių analizės (angl. *context-free*) būdas yra dažniausiai naudojamas sintaksinės analizės būdas. Visi šios algoritmai sukuria hierarchinę junginių struktūrą sakinyje. Tokioje struktūroje mazgai atitinka tam tikro ilgio įėjimo eilutės segmentus, o briaunos nusako, kaip ilgesni segmentai sudaryti iš mažesnių. Viršutinis lygis atitinka visą sakinį. 4 pav. pavaizduota sakinio *„Vilkas yra laukinis žvėris“* frazių struktūra.



Pav. 4. Sudedamųjų dalių struktūra (S – sakinys, DF – daiktavardžio frazė, VF – veiksmažodžio frazė).

Frazių analizėje sakinio vaizdavimui priimta naudoti tokius žymėjimus, sakinys (S) paprastai susideda iš daiktavardinės frazės (DF) ir veiksmažodinės frazės (VF). Daiktavardinę frazę gali sudaryti įvardis (ĮV) arba daiktavardis (D) su kartais pavartojamu būdvardžiu (B). Veiksmažodinę frazę sudaro veiksmažodis (V) ir daiktavardinė frazė (DF) (žr. 5 pav.). Sudėtingesniuose sakiniuose gali būti dar ir prielinksninės konstrukcijos (PK), sudarytos iš prielinksnio (P) ir daiktavardinės frazės (DF) [6].



Pav. 5. Sakinio frazių analizė.

Frazių analizės metodu paremti sintaksiniai analizatoriai yra kuriami turinčioms griežtą žodžių tvarką kalboms, tokioms kaip anglų, vokiečių.

Žymiausi frazių sintaksiniai analizatoriai yra Myiajo (2005) HPSG statistinis sintaksinis frazių analizatorius, M. Collins (2005) CFG sintaksinis analizatorius, E. Charniac (1998) *Edge-based* statistinis sintaksinis analizatorius [12].

Frazių analizės metodas apima šiuos **pagrindinius etapus**:

1. Kiekvienam sakinio žodžiui priskiriama tinkama žodžių klasė.
2. Visi sakinio žodžiai sugrupuojami į sakinio dalių grupes.
3. Kiekvienai grupei nustatoma funkcija (subjektas, objektas ir t. t.).
4. Grupės sujungiamos į sakinį.

Frazių analizės metodo trūkumas yra tas, kad analizė atliekama remiantis žodžių tvarka sakinyje. Todėl šis metodas yra netinkamas lietuvių kalbos sintaksinei analizei [1].

2.4. Problematika

Pagrindinė problema, su kuria susiduriama sintaksinės analizės metu, yra **daugiareikšmiškumas**.

Daugiareikšmiškumas pasireiškia tuo, kad žodžiai, žodžių junginiai ir sakiniai gali būti interpretuojami keliais būdais. Procesas vienam žodžio, žodžių junginio ar sakinio interpretavimo būdui parinkti vadinamas **vienareikšminimu**.

Daugiareikšmiškumo, automatizuojant sintaksinę analizę, šaltiniai yra du: morfologinis daugiareikšmiškumas ir sintaksinis daugiareikšmiškumas.

Morfologinis daugiareikšmiškumas – tai skirtingų žodžių ar žodžio formų sutapimas, pavyzdžiui žodis *padaryti* skirtinguose kontekstuose įgyja skirtingas reikšmes (11) ir (12) [22].

(11) Mano mielas *padaryti*! (daiktavardis, vnsk, Š.)
(12) Jis negali to *padaryti*. (veiksmažodis, bendratis)

Lietuvių kalboje 47 % procentai žodžių yra daugiareikšmiai [22], t. y. turi daugiau nei vieną morfologinę pažymą. Didelis morfologinių pažymų skaičius lemia didesnį galimų analizės medžių skaičių ir taip apsunkina tinkamos medžio reikšmės parinkimą [20].

Church ir Patil 1982 m. nustatė, kad sintaksinėje analizėje galimų medžių skaičius auga drauge su bendru kalbos dalių sakinyje skaičiumi pagal aritmetinę katalonų skaičių priklausomybę (13) [21]:

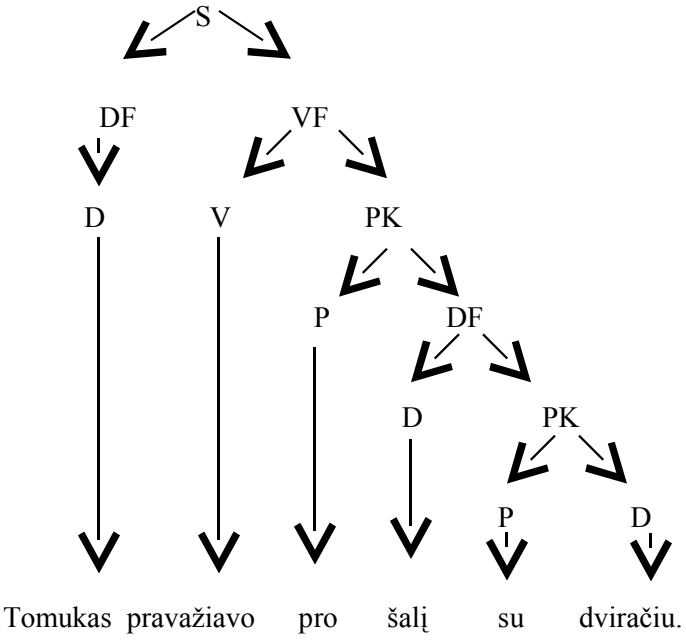
(13) $$C(n) = \frac{1}{n+1}\binom{2n}{n}$$

t. y. jei sakinyje yra 1 kalbos dalis, tai galimų medžių bus skaičius bus max(x) =2, tolimesnės tendencijos pateikiamos toliau:

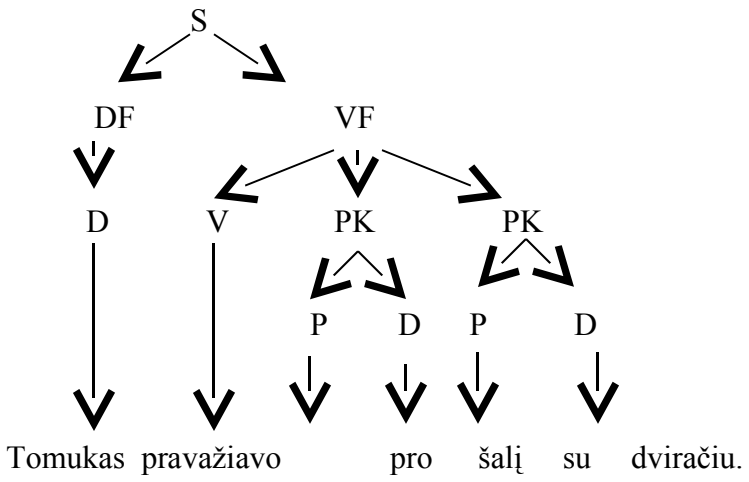
sakinio dalių skaičius		galimų medžių skaičius
1	-	2
2	-	5
3	-	14
4	-	132
5	-	469
6	-	1430

Kitas daugiareikšmiškumo šaltinis sintaksinėje analizėje yra **sintaksinis daugiareikšmiškumas**. Jis atsiranda dėl kelių galimų sakinio semantinių reikšmių. Pavyzdžiui, sakinys „Tomukas pravažiavo pro šalį su dviračiu“ gali būti išanalizuotas dviem būdais (žr. (14) ir (15)).

(14)



(15)



(15) reikšmė yra tokia: Tomukas pravažiavo pro šalį, naudodamas dviratį. (15) reikšmė – Tomukas pravažiavo pro valstybę, kurios atributas dviratis (Šalis su dviračiu).

2.5. Metodo parinkimo pagrindimas

Projekte realizuotas *vientisos* sintaksinių analizatorių strategijos priklausomybių gramatika paremtas metodas. Pagrindiniai kriterijai, nulėmę tokį pasirinkimą, yra šie:

- lietuvių kalba turi laisvą žodžių tvarką, todėl lietuviško sakinio sintaksinei struktūrai pavaizduoti labiau tinka priklausomybių metodas;
- priklausomybių ryšiai yra artimi semantiniams ryšiams, todėl šis metodas vėliau bus naudingas vėlesniuose teksto apdorojimo etapuose;
- priklausomybių medžio vienas mazgas apima vieną žodį. Sintaksinio analizatoriaus darbas yra sujungti egzistuojančius mazgus, o ne išvedinėti naujus (kaip tai daroma frazių gramatikose), todėl priklausomybių gramatikos taikymas sintaksinei kalbos analizei yra tiesiausias kelias;
- priklausomybių gramatika paremti analizatoriai sintaksinę pažymą gali priskirti anksčiau nei frazių gramatika paremti analizatoriai (šiuo būdu atliekama daugiau iteracijų);
- Sistemos veikimo tikslumo gerinimo galimybė;
- Pakartotinis panaudojamumas;
- Išplečiamumas;
- Sistemos palaikymas.

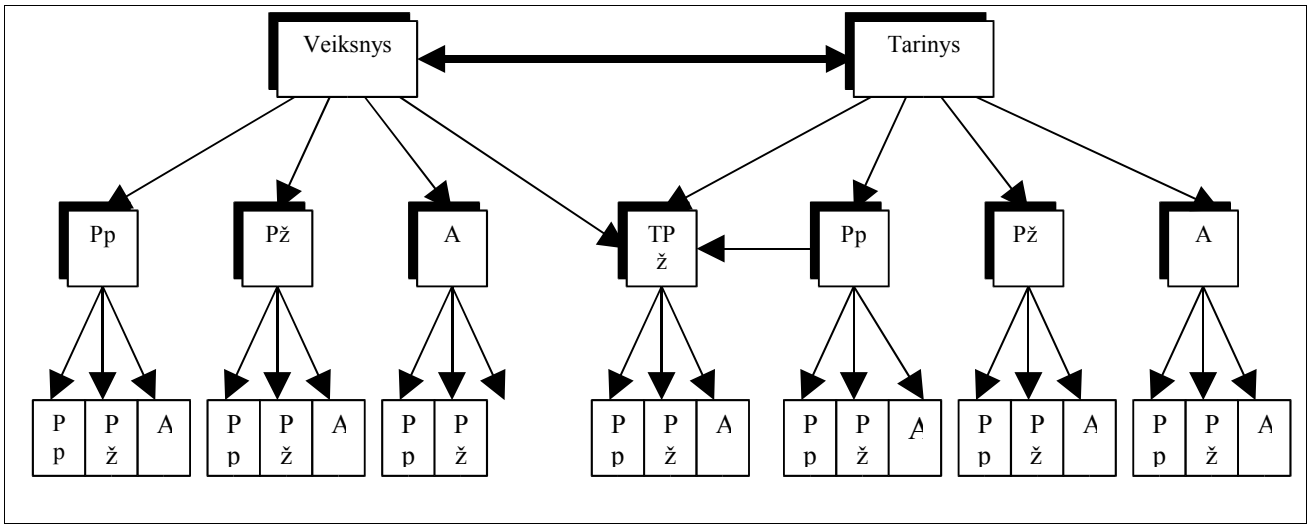
2.6. Aktualių sprendimų apžvalga

2.6.1. Lietuvių kalbos sakinio sintaksinės struktūros pavaizdavimas grafu

D. Šveikauskienė darbe „Lietuvių kalbos sakinio sintaksinės struktūros pavaizdavimas grafu“ (2005) pristato bandymą sudaryti sintaksinės analizės programą, kurioje į vieną visumą būtų sujungtos trys gramatikos sritys: morfologija, sintaksė ir semantika.

Autorė remiasi prielaida, kad lietuvių kalboje sintaksinius ryšius tarp žodžių daugiausia rodo galūnės. Sukurta formalioji gramatika, aprašanti žodžių galūnes, ir jomis paremtus junginius, leidžia generuoti sintaksinius grafus lietuvių kalbos sakiniams. Tokiu būdu sakinio sintaksinės analizės uždavinys yra žodžių junginių ieškojimas ir surastų žodžių junginių sujungimas į grafą [24].

Lietuvių kalbos sakinio gramatikos modelis, kuriuo rėmėsi autorė, atskiroms sakinio dalims pateikiamas 6 pav.



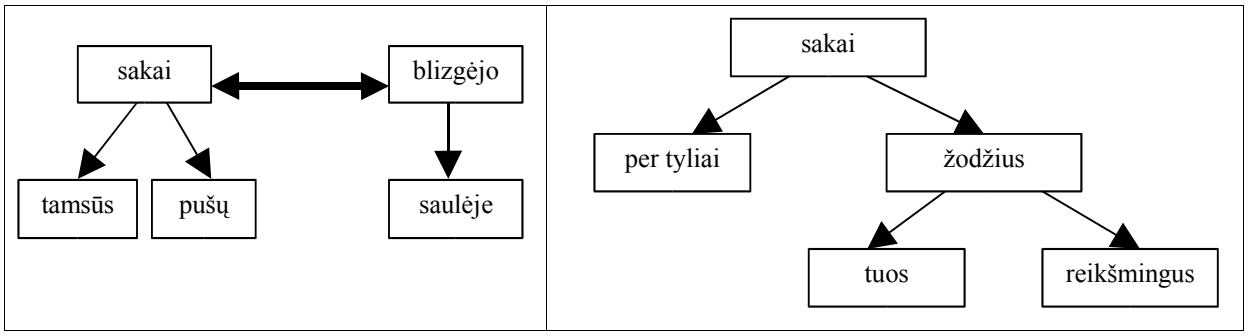
Pav. 6. Apibendrinta vientisinio lietuvių kalbos sakinio struktūrinė schema[1]

Pagrindinės sakinio dalys išdėstomos grafo viršuje, viename lygmenyje ir traktuojamos kaip to paties rango viršūnės. Žemiau pateikiamos antrininkės sakinio dalys, kurios išplečia pagrindines. Pagrindines sakinio dalis nuo antrininkių skiria dar ir tas bruožas, jog visos penkios sakinio dalys (veiksnys, tarinys, papildinys, pažyminys ir aplinkybė) gali būti išplėstos pažyminiu, papildiniu ir aplinkybe. Tačiau nė viena iš jų negali būti išplėsta veiksniu ar tariniu [24].

Sintaksės analizė atliekama remiantis aukščiau paminėtais apribojimais.

[1] Šaltinis: D. Šveikauskienė darbe „Lietuvių kalbos sakinio sintaksinės struktūros pavaizdavimas grafu“ (2005); Pž – pažyminys, Pp – papildinys, A – aplinkybė, TPž – tarininis pažyminys

Analizės metu sudarytų grafų pavyzdžiai sakiniams *„Tamsūs pušų sakai blizgėjo saulėje.“* ir *„Per tyliai sakai tuos reikšmingus žodžius.“* pateikiami 7 pav.



Pav. 7. Sakinių sintaksinės struktūros[2]

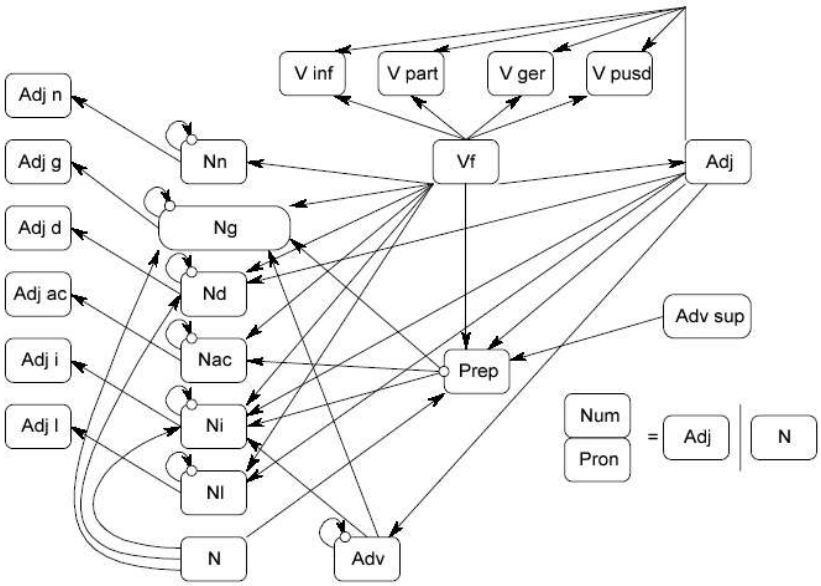
Šis sprendimas dar nėra iki galo įgyvendintas, todėl nežinomas siūlomos metodikos pritaikymo tikslumas.

2.6.2. Lietuvių kalbos sakinio sintaksinė analizė

V. Daudaravičiaus darbe „Lithuanian Sentence Syntax Analysis“ (2002) aprašoma lietuvių kalbos formalizavimo sintaksė, siūlomi žymėjimai kai kuriems sintaksės grafo generavimo apribojimams (laisva žodžių tvarka sakinyje, neprojektiškumas, priklausomybių savybės). Sudarant sintaksines struktūras autorius naudoja grafus.

Lietuvių kalbos priklausomybių gramatikos modelis vaizduojamas priklausomybių grafu atskiros kalbos dalims pateikiamas 8 pav.:

[2] Šaltinis: D. Šveikauskienė darbe „Lietuvių kalbos sakinio sintaksinės struktūros pavaizdavimas grafu“ (2005);



Pav. 8. Priklausomybių gramatikos modelis lietuvių kalbos gramatikai[3]

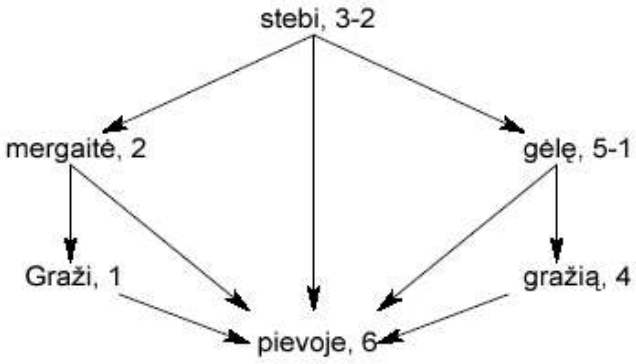
Autoriaus aprašomas būdas siūlo sintaksinį analizatorių lietuvių kalbai kurti **dviem etapais**:

- Panaudojant tam tikrus priklausomybių gramatikos apribojimus, ***sugeneruoti priklausomybių grafą***. Ši dalis reikalauja sintaksės kalbos žinių, kadangi pagal formalias priklausomybių taisykles kuriami ryšiai tarp gramatiškai susijusių žodžių bei generuojamas priklausomybių medis.
- „***Ištraukti priklausomybių medį iš priklausomybių grafo***“ [3]. Šioje dalyje, remiantis projektiškumo apribojimais, iš priklausomybių grafo išrenkami visi galimi priklausomybių medžių variantai.

Minėtus etapus autorius vadina *lingvistiniu* (pirmas etapas) lygmeniu ir *techniniu* (antrasis etapas) lygmeniu.

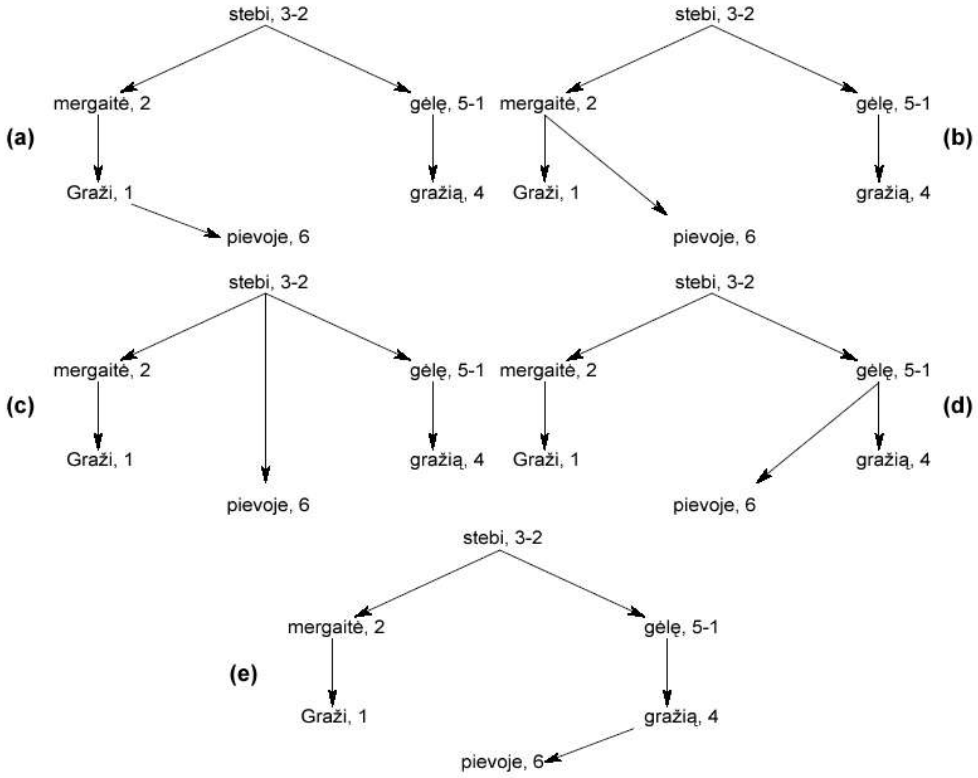
Priklausomybių grafo, kuris generuojamas pirmame analizatoriaus įgyvendinimo etape, pavyzdys pateikiamas 9 pav.

[3] Šaltinis: V. Daudaravičius „Lietuvių kalbos sakinio sintaksinė analizė“, 2002.



Pav. 9. Priklausomybių grafas sakiniui *Graži mergaitė stebi gražią gėlę pievoje*.[4]

Būdai kaip iš anksčiau pateikto priklausomybių grafo išrenkami priklausomybių medžiai nagrinėjamam sakiniui *Graži mergaitė stebi gražią gėlę pievoje*, vaizduojami 10 pav.



Pav. 10. (a):(e) – galimi priklausomybių medžiai sakiniui *Graži mergaitė stebi gražią gėlę pievoje*.[5]

[4] Šaltinis: V. Daudaravičius „Lietuvių kalbos sakinio sintaksinė analizė“, 2002
[5] Šaltinis: V. Daudaravičius „Lietuvių kalbos sakinio sintaksinė analizė“, 2002

3. PROJEKTINĖ DALIS

Projekto tikslas – sukurti sintaksinės analizės modulį, paremtą priklausomybių gramatika, kuris leistų atlikti automatinį lietuvių kalbos apdorojimą, būtų integruojamas į kitus kalbos analizės įrankius.

Procesorius, kaip atskiras objektas, eiliniam informacinės visuomenės vartotojui nėra reikalingas ar labai naudingas, tačiau procesorius, kaip sudėtinė kompiuterio dalis įgyja visai kitą prasmę – be jo kompiuteris negali funkcionuoti. Panaši yra realizuoto įrankio – dalinio **sintaksinio analizatoriaus** (paremto **priklausomybių gramatika**) nauda: atskirai šis įrankis tinkamas tik siaurai specializuotai vartotojų sričiai (pvz., lingvistams analizuojantiems sakinio struktūras), tačiau be šio įrankio nėra galimi jokie darbai, kuriems reikalinga teksto sintaksinė analizė. Tokie darbai gali būti: automatinis gramatikos tikrintuvas, informacijos iš teksto išgavimo įrankis, automatinė tekstų vertimo iš vienos kalbos į kitą sistema ir kt.

Besivystant lietuvių kalbos technologijai, itin aktualus poreikis šiam įrankiui. Sintaksinis analizatorius būtinas tolimesnių kalbos lygių analizei (pavyzdžiui, semantika).

Projekto iniciatorius, užsakovas ir būsimas vartotojas yra VDU Kompiuterinės lingvistikos centras.

Sistemos reikalavimams specifikuoti naudojamas CASE įrankis *Magic Draw UML 9.5*, projekto procesui aprašyti pasirinkta RUP projektavimo metodika.

3.1. Sistemos paskirtis

Šis projektas yra vienas iš pirmųjų priklausomybių gramatikos taikymų lietuvių kalbos automatizuotoje analizėje. Pilnos sintaksinės automatinės analizės įrankių lietuvių kalbai nėra sukurta, o *be* ***sintaksinio analizatoriaus*** *negalimas* ***automatizuotas tekstų apdorojimas****, kuriam reikalinga* ***sintaksinė analizė*** [10].

Natūrali ar dirbtinė kalba turi tam tikrą apibrėžtą struktūrą, vadinamą sintakse. Esminė sintaksės idėja yra tai, kad žodžiai tarpusavyje yra priklausomi ir gali būti sugrupuoti į sakinio dalis taip, kad sudarytų sakinį. Formaliai aprašius šias struktūras, sukurtas įrankis nustatantis sakinio sintaksines struktūras. Pristatomo **sintaksinio analizatoriaus** veikimo principas aptariamas skyrelyje 3.3.

3.2. Pagrindiniai reikalavimai

3.2.1. Sistemos vartotojai

Sistema tinkama naudoti vartotojams, turintiems nedidelį kompiuterinio raštingumo lygį. Sistemos turi tokius aktorius:

Vartotojas – tai asmuo, kuris naudos sistemą sintaksinei analizei atlikti. Jis gali pasirinkti analizės tekstą ir peržiūrėti ir saugoti analizės rezultatus.

Lingvistas – tai asmuo, kuris gali papildyti taisyklių sąrašą naujomis ir atlikti visas *vartotojo* funkcijas.

3.2.2. Vartotojo sąsajos reikalavimai

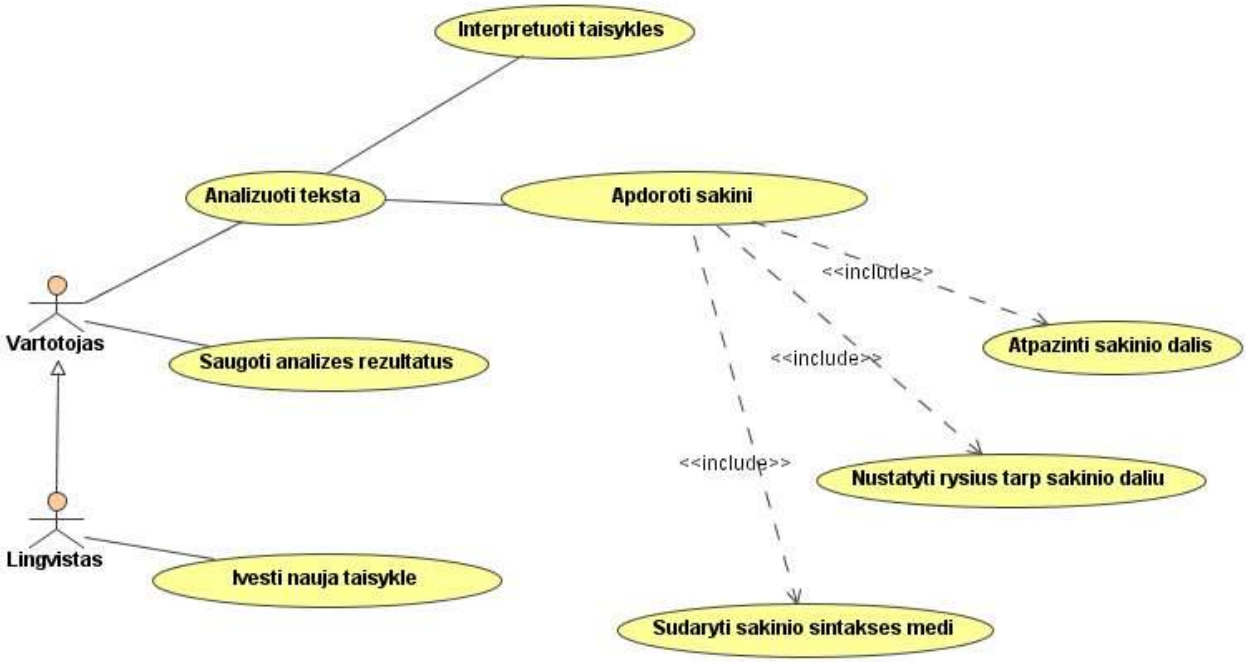
Vartotojo sąsaja turi būti kuriama taip, kad ji būtų įprasta, suprantama ir lengvai įsisavinama [15]. Taip pat svarbu, kad vartotojo sąsaja užtikrintų teisingą sistemos vartojimą ir apsaugotų nuo klaidų įvėlimo. Toliau pateikiami bendri reikalavimai vartotojo sąsajai:

- Veiksmai turi būti nuspėjami ir intuityvūs – tai leidžia vartotojams lengvai naudotis sistema.
- Veiksmai, kurių vartotojas tuo momentu atlikti negali, turi būti neleistini – tai aktualu numatytoje veiksmų sekoje. Pavyzdžiui, nenuskaičius sakinio, negalima sintaksinė analizė.
- Operacijos turi būti efektyvios ir greitos, vengiama bereikalingų veiksmų. Pasikartojantys veiksmai gali būti apjungiami.
- Sistema turi būti bendraujanti – vartotojas turi susilaukti pranešimų apie sėkmingą operacijos įvykdymą, paklausimų prieš išeinant iš sistemos ir pan.

Vartotojo sąsajos langai pateikiami 1 priede.

3.2.3. Reikalavimai sistemai

Funkcionalumo reikalavimus sistemai atitinka panaudos atvejų diagrama (PA). Sistemos įgyvendintų PA diagrama pateikiama 11 pav.



Pav. 11. Sistemos panaudos atvejų diagrama.

Įgyvendinti panaudos atvejai yra šie:

- *Analizuoti tekstą*. Sistema analizuoti pateikiamą tekstą. Tai pagrindinė sistemos funkcija.
- *Interpretuoti taisykles*. Sistema interpretuoja taisykles, jos reikalingos sintaksinei analizei.
- *Apdoroti sakinį*. Šis panaudos atvejis susideda iš *sudaryti sakinio sintaksės medį*, *nustatyti ryšius tarp sakinio dalių*, *atpažinti sakinio dalis* panaudos atvejų.
- *Atpažinti sakinio dalis*. Sistema apdoroja pateikiamą tekstą, pasinaudoja tekste esančia informacija ir parenka sakinio žodžiams jų funkcijas (pavyzdžiui, nustato predikatą, subjektą).
- *Nustatyti ryšius tarp sakinio dalių*. Sistema pasinaudoja sintaksės taisyklėmis ir tekste esančia morfologine informacija ir nustato sakinio žodžius siejančius ryšius (pavyzdžiui, junginys: predikatas → subjektas).
- *Sudaryti sakinio sintaksės medį*. Sistema iš sudarytų junginių sukonstruoja sintaksės medį taip, kad į analizės medį būtų įtraukiami visi sakinio žodžiai.
- *Saugoti analizės rezultatus*. Sistema leidžia išsaugoti analizės rezultatus.

- *Įvesti naują taisyklę*. Sistema turi galimybę taisyklių DB papildyti nauja sintaksės taisykle.

Pagrindiniai nefunkciniai reikalavimai:

- Įprasta, intuityvi sąsaja ir meniu;
- Sistema turi neleisti vartotojui daryti klaidų;
- Visapusiška pagalba vartotojui;
- Sistema turi būti lengvai išplečiama papildomais komponentais, papildomu funkcionalumu.

3.2.4. Reikalavimai duomenims

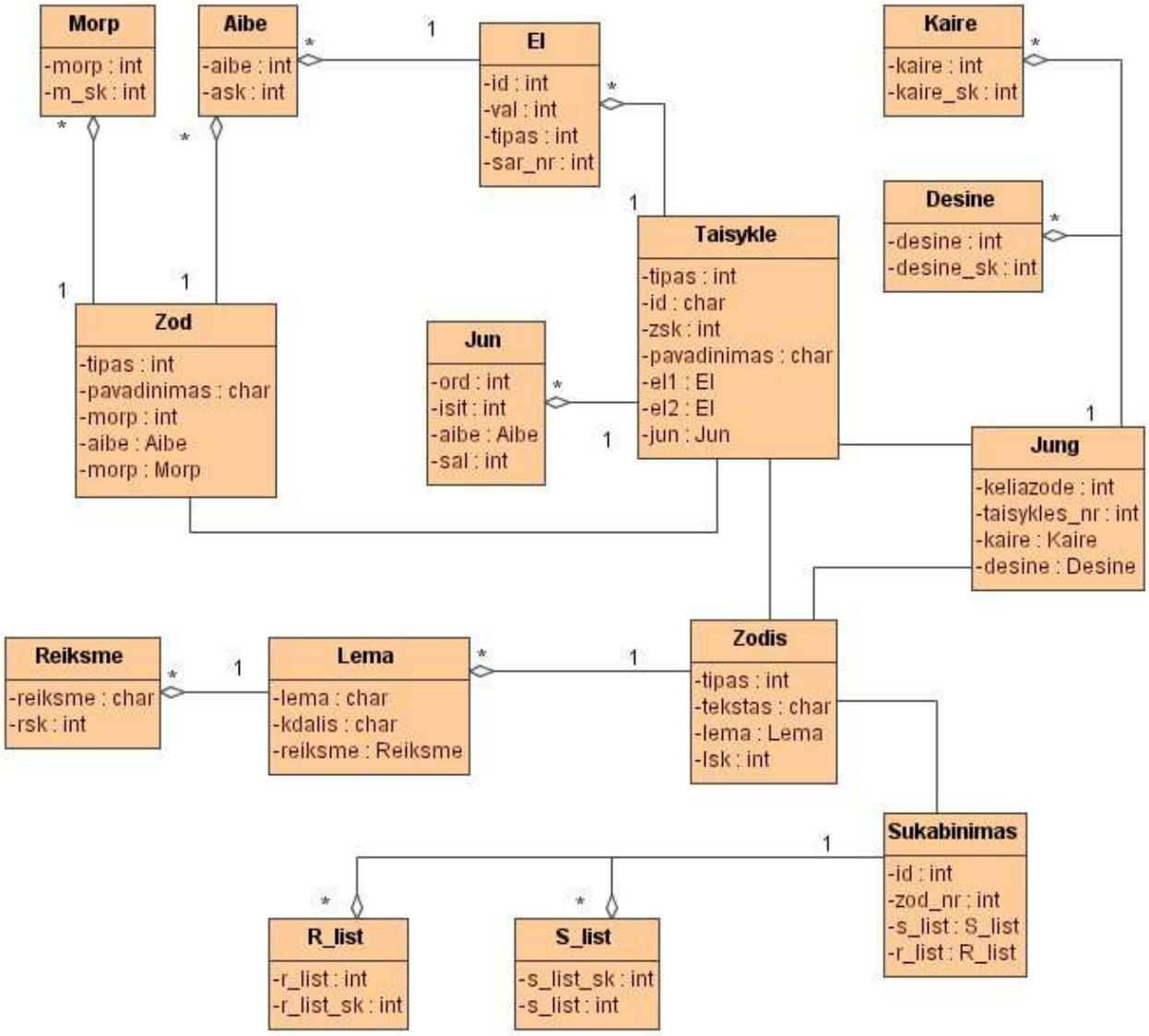
Kuriant sintaksinį analizatorių, būtinos šios pagrindinės duomenų esybės:

- Nagrinėjamas žodis;
- Taisyklė;
- Junginys;
- Sakinio medis.

Kalbose, kuriose žodžiai yra kaitomi, gramatikos taisyklių skaičius gali labai išaugti. Todėl patogu naudoti sudėtines kategorijas, sudarytas iš atributų ir jų reikšmių aibių, taip pat mechanizmo, leidžiančio patikrinti, ar suderinami dviejų skirtingų elementų atributai. Reikšmių atributams priskyrimas ir suderinamumo tikrinimas vadinamas unifikavimu [21]. Šiuo atveju galima rašyti apibendrintas taisykles, pvz., kad du elementai turi būti suderinti skaičiumi ir linksniu. Konkrečios šių atributų reikšmės įstatomos suderinamumo tikrinimo metu. Be to, suderinamumo reikalavimas gali būti leidžiamas abiem kryptimis neribotu atstumu.

Atsižvelgiant į šias kalbos savybes, buvo suprojektuoti ir realizuoti sintaksiniam analizatoriui būtini duomenų tipai.

Pagrindiniai reikalavimai duomenims pateikiami duomenų modelio schemoje 12 pav. Duomenų modelyje pavaizduoti pagrindinių duomenų – *taisyklių*, *analizuojamų žodžių*, *sukabinimų* ir *junginių* naudojami formatai.

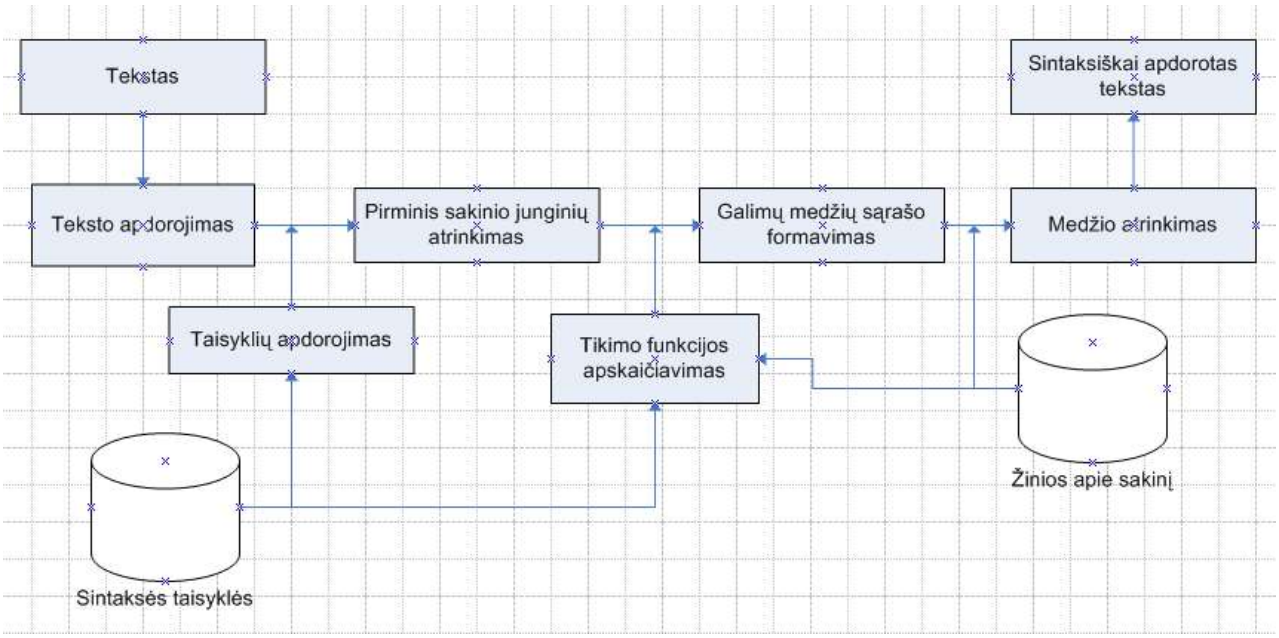


Pav. 12. Duomenų modelio schema.

Sistemos naudojamų duomenų pavyzdžiai pateikiamas 2 priede.

3.3. Sistemos architektūra

Pristatomo **sintaksinio analizatoriaus** veikimo schema pateikiama 13 pav.



Pav. 13. Automatinės sintaksinės analizės blokinė schema

Realizuota šio darbo dalis apima *teksto apdorojimą*, *taisyklių apdorojimą*, *pirminį sakinio junginių atrinkimą*, *tikimo funkcijos apskaičiavimą*, *galimų medžių sąrašo formavimą*. Esant geriausiam atvejui, t. y. kuomet *galimų medžių sąraše* yra tik vienas medis, gaunamas *sintaksiškai apdorotas tekstas*. Jei *galimų medžių sąraše* yra daugiau nei vienas medis, analizės rezultate pateikiami keli medžiai.

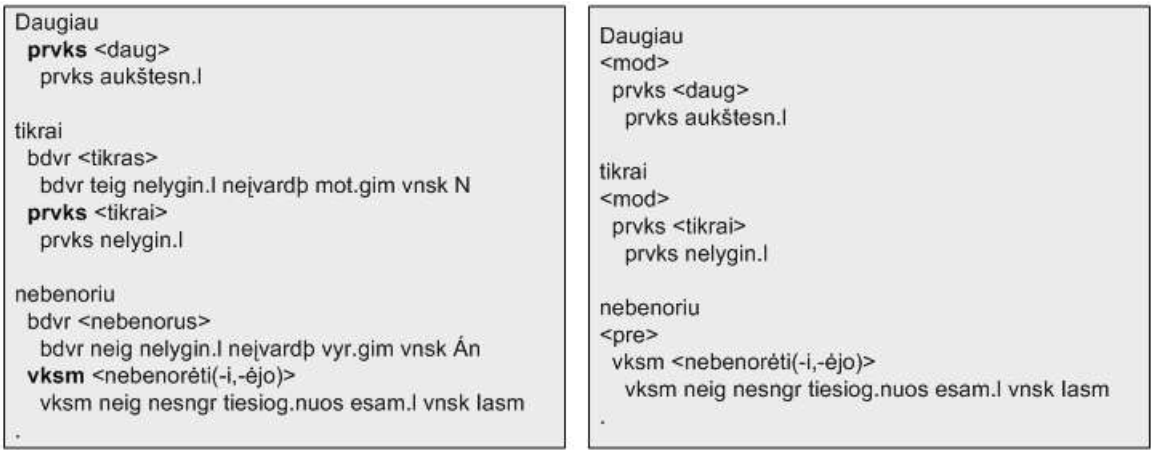
Automatinės sintaksinės analizės sistema susideda iš šių dalių:

- *Teksto apdorojimas*. Sakinio žodžių informacijos parengimas analizei (žodžių, lemų ir morfologinių pažymų atskyrimas).
- *Taisyklių apdorojimas*. Taisyklių informacijos parengimas analizei. Taisyklių apdorojimo resursas yra *sintaksinės taisyklės*.
- *Pirminis sakinio junginių atrinkimas*. Šiam etapui naudojami *teksto* ir *taisyklių* apdorojimo rezultatai. Sistema įvertina, kurios taisyklės gali būti pritaikytos kuriems sakinio žodžiams. Tokiu būdu iš sakinio žodžių ir juos jungiančių taisyklių sukuriamas junginys. Peržiūrėjus visas turimas taisykles, suformuojamas sakinio žodžių junginių sąrašas.
- *Galimų medžių sąrašo formavimas*. Iš prieš tai buvusiame etape suformuoto sakinio žodžių junginių sąrašo, pritaikius *tikimo funkciją,* formuojami medžiai. Algoritmas įgyvendinamas iteraciniu principu, t. y. jis tęsiamas tol, kol iš atskirų žodžių junginių

suformuojamas sakinys. Šiame etape naudojamas greičiausio nusileidimo (angl. *hill-climbing*) lokalios paieškos algoritmas.

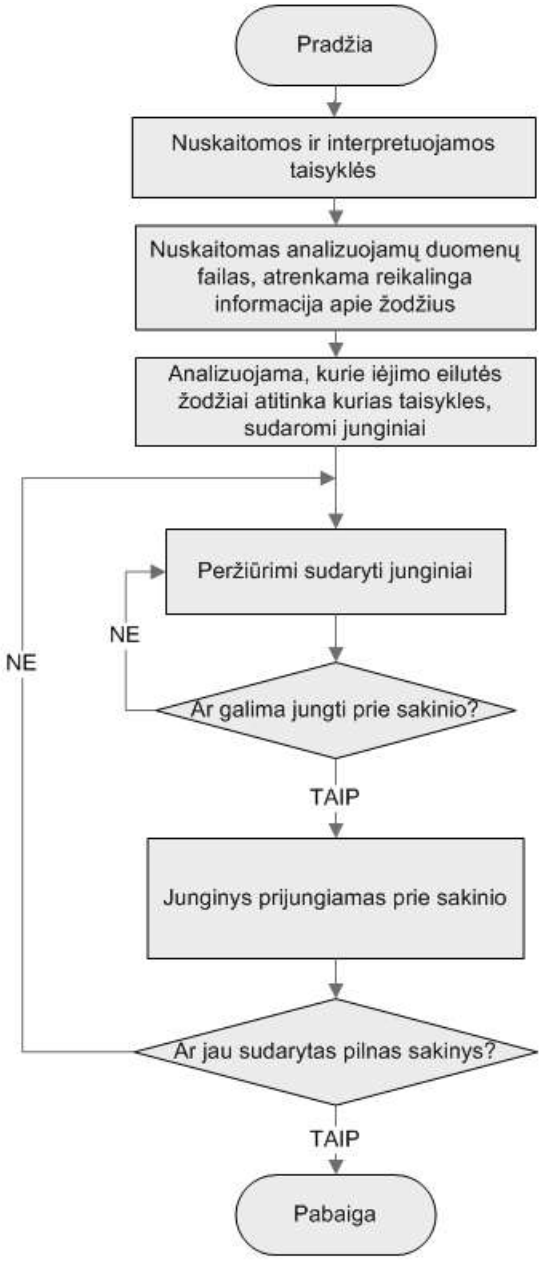
- *Tikimo funkcijos apskaičiavimas*. Tikimo funkcija apskaičiuojama remiantis *sintaksės taisyklėmis* ir *žiniomis apie sakinį*. Tikimo funkcija grąžina įvertį, nusakantį junginio tikimą formuojamam sakiniui.
- *Medžio atrinkimas.* Jei analizuojamas sakinys turi daugiau nei vieną sprendinį, pasinaudojus *žiniomis apie sakinį,* parenkamas vienas tinkamiausias medis.
- Analizės proceso rezultatas yra *sintaksiškai apdorotas tekstas*.

Sintaksiškai apdoroto teksto (išėjimo duomenys) ir *teksto* (įėjimo duomenys) pateikiami pav.14.



Pav. 14. Įėjimo ir išėjimo duomenys.

Sintaksinės analizės proceso algoritmas pateikiamas 15 pav.

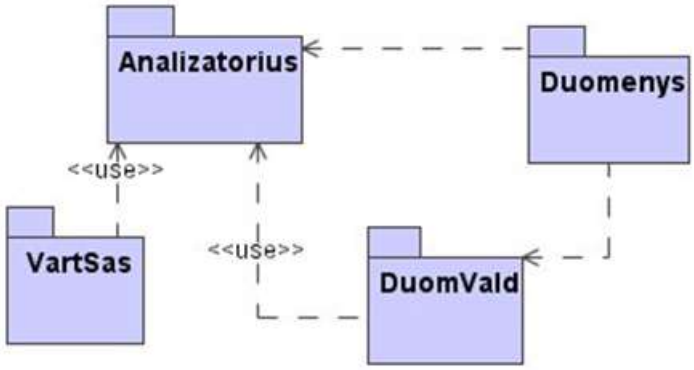


Pav. 15. Sintaksinės analizės proceso algoritmas.

Sistemai pasirinktas architektūros modelis vaizduojamas 16 pav. Pagal šį modelį, priklausomybių gramatika paremto sintaksinio analizatoriaus sistemą sudaro šie paketai:

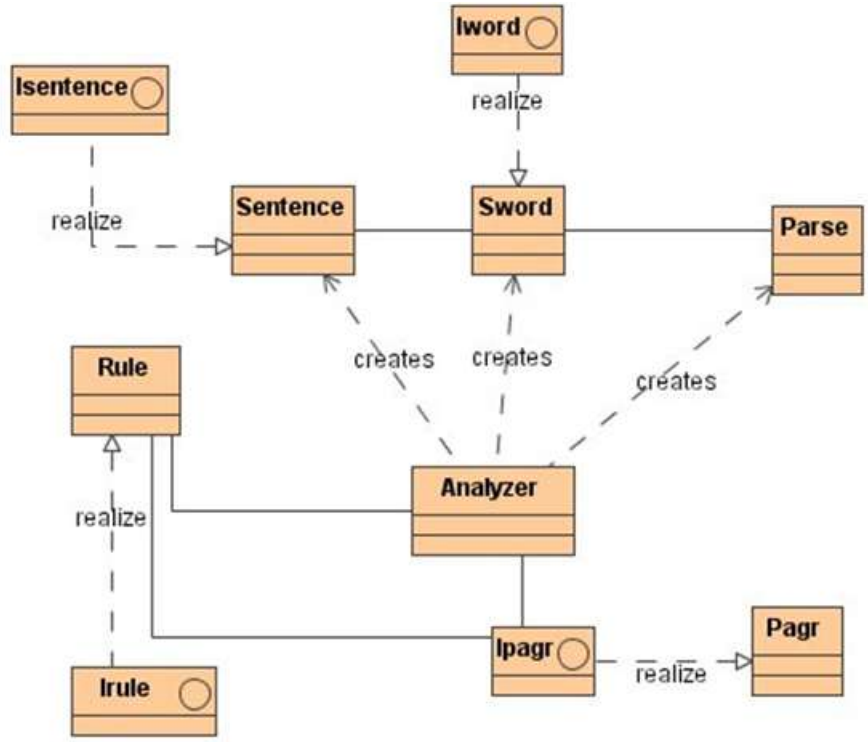
- „Analizatorius“ – pagrindinis analizatoriaus paketas;
- „VartSas“ – grafinę vartotojo sąsają realizuojantis paketas;
- „DuomVald“ – sistemos duomenų iškvietimo, siuntimo, saugojimo, valdymo paketas;

- „Duomenys“ – sistemos apdorojamų duomenų paketas.



Pav. 16. **Sistemos išskaidymas paketais**

Architektūrinis sprendimas padengia kuriamos sistemos paketus. Paketas „Analizatorius“ atlieka svarbiausias sistemos funkcijas, jo klasių diagrama pateikiama 17 pav.:

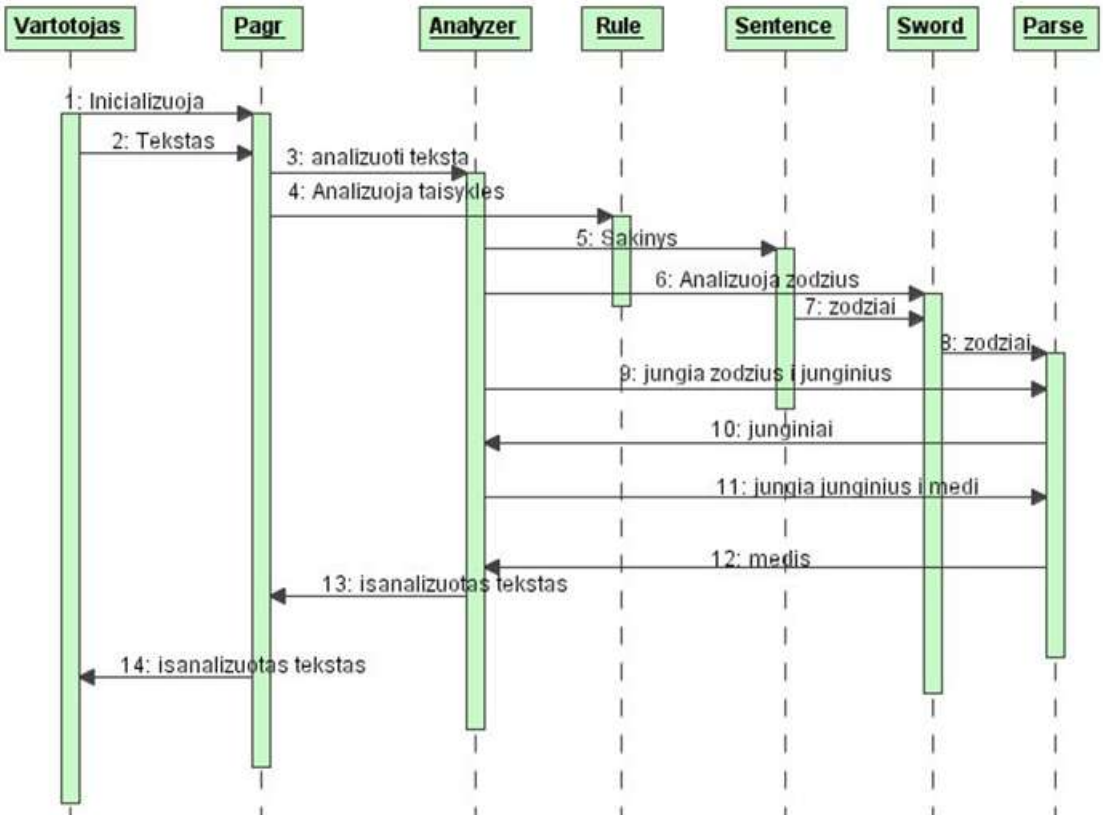


Pav. 17. Paketo „analizatorius“ detalizavimas

Dinaminiam sistemos vaizdui atskleisti toliau pateikiamos esminės paketo *analizatorius* sekų, būsenų, bendradarbiavimo diagramos.

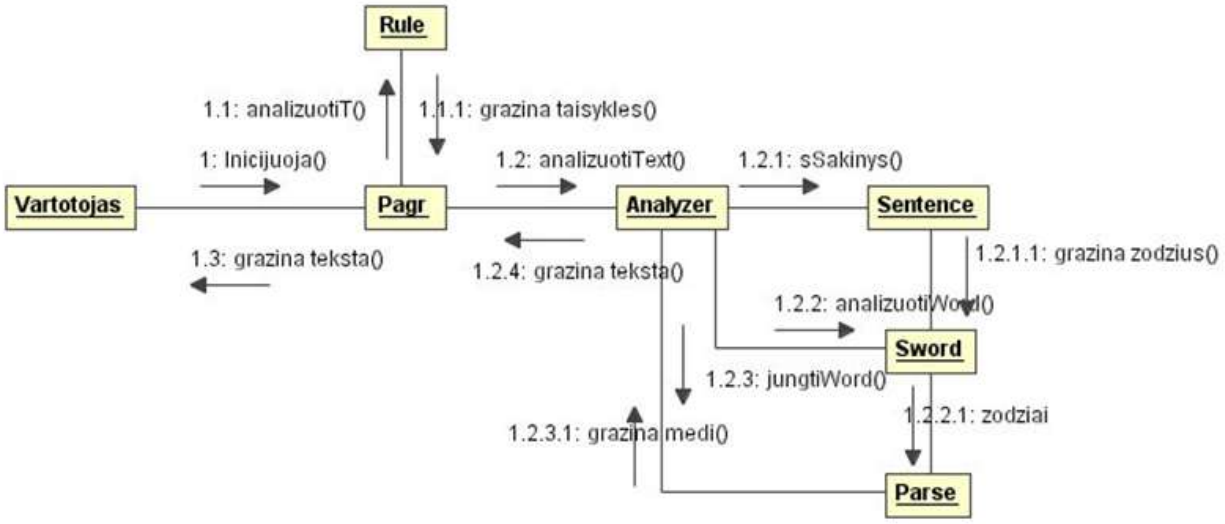
Paketas *analizatorius* yra atsakingas už sakinio sintaksinę analizę. Jo duomenys atlieka sakinio žodžių analizę, junginių pagal taisykles formavimą ir sakinio sintaksės medžio konstravimą.

Sekų diagramoje (18 pav.) vaizduojamas sintaksinės analizės procesas: Vartotojas inicijuoja sistemą, pateikia tekstą analizei – 1 ir 2 žingsniai. 3 – 7 žingsniuose sistema analizuoja turimas taisykles ir gautą tekstą (įėjimo duomenys). 8 ir 9 žingsnyje pagal turimas taisykles nustatomi sakinio žodžių junginiai. 10 ir 11 žingsniuose junginiai yra „segami“ į analizės medį. 12 žingsnis – analizės medis gražinamas *analyzer* ir perverčiamas į vartotojui suprantamą formatą. 13 ir 14 žingsniuose išanalizuotas tekstas su sintaksės pažymomis grąžinamas vartotojui.



Pav. 18. Sintaksinės analizės proceso sekų diagrama.

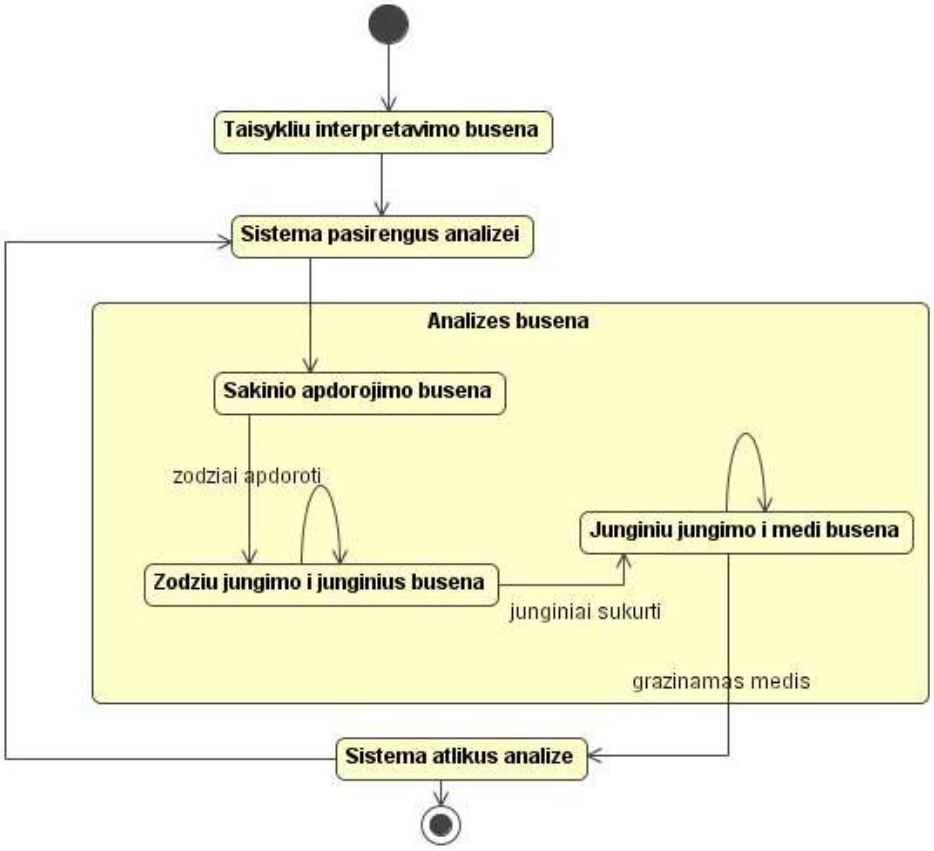
Aiškesniam sintaksinės analizės proceso vaizdui sukurti, pateikiama bendradarbiavimo diagrama (žr. 19 pav.).



Pav. 19. Sintaksinės analizės proceso bendradarbiavimo diagrama

Sistemos sintaksinės analizės proceso būsenų diagramoje (20 pav.) vaizduojamos būsenos, į kurias gali pakliūti sistema:

- *Taisyklių interpretavimo būsena* – į šią būseną patenkama pasirinkus taisyklių interpretavimo veiksmą. Ši būsena būtina prieš *analizės būseną*.
- *Sistema pasirengusi analizei* – tai pasirengusios atlikti analizės darbą, pagal turimas taisykles, sistemos būsena.
- *Analizės būsena* – tai būsena, į kurią sistema patenka po analizės vykdymo funkcijos iškvietimo. Ši būsena apima *sakinio apdorojimo būseną*, *žodžių jungimo į junginius būseną*, *junginių jungimo į medį būseną*:
  - *Sakinio apdorojimo būsena* – tai vidinė analizės būsena.
  - *Žodžių jungimo į junginius būsena* – vidinė analizės būsena, kuri tęsiasi tol, kol visiems sakinio žodžiams patikrinamos jungimosi taisyklės.
  - *Junginių jungimo į medį būsena* – vidinė analizės būsena, kuri tęsiasi tol, suformuojamas analizės medis.
- *Sistema atlikus analizę* – tai atlikusios analizę, sistemos būsena. Iš šios būsenos galima grįžti į *sistema pasirengusi analizei* būseną arba išeiti iš sistemos.



Pav. 20. Sintaksinės analizės proceso būsenų diagrama.

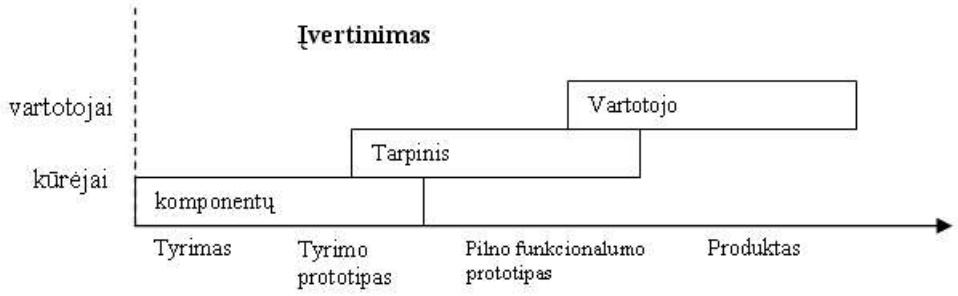
Sistema įgyvendinta C++ kalba. Taisyklės ir analizuojami duomenys yra tekstiniuose failuose.

Sistemos veikimas buvo ištestuotas Windows platformos XP, 2000, 98 ir 95 versijose.

4. TYRIMO DALIS

Keičiantis ir tobulėjant kalbų technologijos sričiai, pamažu pereinama prie komercinių produktų gamybos ir kalbų technologijos programinės įrangos įvertinimas tampa vis svarbesnis.

Įvertinimo technologijos pasirinkimas glaudžiai susijęs su programinės įrangos procesu. 21 pav. pavaizduotos keturios apibendrintos programinės įrangos gyvavimo ciklo stadijos: tyrimas, tyrimo prototipas, funkcionuojantis prototipas, produktas galimos įvertinimo metodikos [21].



Pav. 21.Technologinis gyvavimo ciklas ir įvertinimas

Realizuota sistema užima tarpinę vietą sistemos gyvavimo ciklo skalėje – tarp tyrimo prototipo ir pilno funkcionalumo prototipo, todėl tinkamiausia įvertinimo strategija yra komponentų įvertinimo strategija.

Sistemos įvertinimo būdai taip pat priklauso nuo sistemos pobūdžio, įėjimo ir išėjimo duomenų. Remiantis šiais aspektais, skiriamos analizės sistemos, generuojančios sistemos ir interaktyvios sistemos.

Realizuota sistema yra analizės sistema. Jai būdinga tai, kad įėjimo duomenys yra natūralios kalbos tekstas, o išėjimo duomenys – abstraktus įėjimo duomenų vaizdavimas. Tokioms sistemoms taikomos kelios įvertinimo metodikos [12]:

- Savybėmis pagrįstų metrikų įvertinimas – apibrėžiamos aktualios sistemos savybės ir tikrinama ar sistema jomis pasižymi.
- Lyginimas su ekvivalentiškomis sistemos – nustatomi kriterijai, parodantys svarbiausias sistemos savybes ir lyginama keletas sistemų.
- *Gold standard* (auksinio standarto) įvertinimas – „rankiniu“ būdu sukuriamas „teisingų“ išėjimo duomenų rinkinys ir palyginamas su sistemos sukurtais išėjimo duomenimis. Procentinis įvertis – veikimo tikslumas (angl. *accuracy*) [17].

4.1. Sistemos kokybės įvertinimas

Savybėmis pagrįstų metrikų įvertinimas – pirmoji taikyta sukurtos sistemos įvertinimo metodika. Tyrimo metu buvo vertinama sukurtos programinės įrangos, skirtos sintaksinei analizei automatizuoti, kokybė. Tuo tikslu buvo pasirinkta dešimt kokybės vertinimo kriterijų. Šios tyrimo dalies rezultatų suvestinė pateikiama 1 lentelėje.

Lentelė. 1. Programinės įrangos vertinimo rezultatai

Kokybės kriterijus	Sistemos įvertinimas
Korektiškumas	Atitinka. Realizuotos sistemos funkcijos veikia korektiškai.
Patikimumas	Atitinka. Testavimo laikotarpiu nebuvo pastebėta sistemos gedimų.
Efektyvumas	Atitinka. Didžiausias krūvis sistemai tenka sprendžiant perrinkimo uždavinius. Darbo greičiui pagerinti buvo optimizuoti sistemos duomenų tipai, naudojamas indeksavimas, taikytas greito nusileidimo lokalios paieškos algoritmas.
Integralumas	Kol kas nėra. Numatoma ateityje sistemą plėsti taip, kad būtų užtikrintas duomenų integralumas, t. y. realizuoti taisyklių įvedimo sąsają, kuri apsaugotų nuo pasikartojančių duomenų ir klaidų įvedant naujus duomenis.
Saugumas	Nėra. Programoje nėra realizuota vartotojų autorizavimo galimybė.
Panaudojamumas	Atitinka. Sistema yra paprasta naudoti, lengvai įsisavinama.
Išplečiamumas	Atitinka. Sistema buvo kuriama remiantis programų kūrimo procesu, yra pilnai dokumentuota.
Pernešamumas	Atitinka. Sistema įgyvendinta C++ programavimo kalba, todėl yra nepriklausoma nuo platformos.
Sąsajos galimybės	Kol kas nėra. Numatoma ateityje sistemą plėsti realizuoti sąsajas su *Dabartinės lietuvių kalbos tekstyno* duomenų baze.
Pakartotinis panaudojamumas	Atitinka. Sistema buvo kuriama remiantis programų kūrimo procesu, yra pilnai dokumentuota.

4.2. Sistemos įvertinimas panašių sistemų atžvilgiu

Lyginimas su ekvivalentiškomis sistemos – antroji taikyta sukurtos sistemos įvertinimo metodika.

Šio darbo analitinėje dalyje buvo pristatytos panašios į autorės realizuotą sistemos: „Lietuvių kalbos sakinio sintaksinės struktūros pavaizdavimas grafu“ (2005) ir „Lietuvių kalbos sakinio sintaksinė analizė“ (2002). Norint pateikti įvairiapusiškesnį sistemos įvertinimą, buvo nuspręsta palyginti šias sistemas su realizuotu sintaksiniu analizatoriumi.

Sistemas galima palyginti teoriniu aspektu, kadangi realiai jos nėra naudojamos sintaksinei analizei atlikti. Sistemų palyginimo suvestinė pateikiama 2 lentelėje.

Lentelė. 2. Sintaksinės analizės įrankių palyginimas

Įrankis Kriterijus	Lietuvių kalbos sakinio sintaksinė analizė (2002)	Lietuvių kalbos sakinio sintaksinės struktūros pavaizdavimas grafu (2005)	Priklausomybių gramatikos taikymas lietuvių kalbos apdorojime (2006)
Taikyta sintaksinės analizės strategija	Iš apačios į viršų	Iš apačios į viršų	Iš apačios į viršų
Taikyta sintaksinės analizės metodika	Priklausomybių gramatika	Priklausomybių gramatika, frazių gramatika	Priklausomybių gramatika
Apribojimų panaudojimas	Yra	Yra	Yra
Sintaksės taisyklių panaudojimas	Nėra	Nėra	Yra
Morfologijos įtraukimas	Yra	Yra	Yra
Semantikos taisyklių panaudojimas	Nėra	Yra (susiję su taikytais apribojimais)	Nėra
Grafo/priklausomybių medžio formavimas	Yra (formuojamas grafas)	Yra (formuojamas grafas)	Yra (formuojamas medis)
Grafo/priklausomybių medžio vaizdavimas	Yra	Nėra	Nėra
Atsižvelgimas į kalbos dalis	Yra	Yra	Yra
Atsižvelgimas į sakinio dalis	Nėra	Yra	Yra
Projektiškumo kriterijus	Yra	Nėra	Yra
Įsiterpimo kriterijus	Yra (projektiškumas)	Yra	Yra

Laisvos žodžių tvarkos kriterijus	Yra	Yra	Yra
Valdymo (prisijungimo) kriterijus	Yra	Yra	Yra
Veikimo principas	Generuojami visi galimi sakinio sintaksiniais ryšiais sujungti grafai, naudojamas projektiškumo kriterijus tinkamiems grafams atrinkti.	Daroma prielaida, kad sakinyje yra veiksnys ir tarinys, pagal tai sugeneruojami visi sakinio analizės grafai, vėliau bandoma atrinkti tinkamus.	Remiantis taisyklėmis ir apribojimais sugeneruojami galimi sakinio analizės variantai, jei tokių variantų ne vienas, atliekamas tinkamiausio nustatymas.
Veikimo tikslumas	Nežinomas	Nežinomas	80,2 % (tirta atskirų sakinio dalių nustatymo tikslumas)

Iš 2 lentelės matyti, kad tris sprendimus vienija priklausomybių gramatika ir pasirinkta sintaksinės analizės strategija. Dėl šios priežasties visose sistemose randami panašumai, pavyzdžiui, atsižvelgimas į morfologiją, kalbos dalis, apribojimų panaudojimą. Esminiai skirtumai, išskiriantys realizuotą sintaksinį analizatorių, yra sintaksinių taisyklių rinkinio panaudojimas ir veikimo tikslumas, vertinant atskirų sakinio dalių atpažinimo vidurkį ir pats veikimo principas (algoritmas).

4.3. Sistemos Gold standard įvertis

Gold standard (auksinio standarto) įvertinimas – tai trečioji taikyta sukurtos sistemos įvertinimo metodika.

Tyrimo metu buvo vertinama sukurtos programinės įrangos, skirtos sintaksinei analizei automatizuoti, **išėjimo duomenų atitikimas „rankiniu“ būdu sukurtiems sintaksiškai išanalizuotiems sakiniams**. Realizuota sistema ne visais atvejais pateikia vieną sintaksinės analizės sprendimą. Todėl užuot vertinus atitikimą sakinių lygmenyje, pasirinktas atitikimo vertinimas sakinio dalių (objektas, subjektas, predikatas, atributas, aplinkybė) lygmenyje.

Pirmiausia tuo tikslu buvo išanalizuota 20 lietuvių kalbos sakinių [18], t. y. nustatytos kiekvieno sakinio kalbos dalys ir „rankiniu“ būdu sudėtos jas pažyminčios sintaksinės pažymos.

Pavyzdžiui, sakinys „Saulelė krypo vakarop“ su visomis galimomis morfologinėmis pažymomis pateikiamas 3 lentelės kairėje pusėje, o su rankiniu būdu sudėtomis sintaksinėmis pažymomis – dešinėje pusėje.

Lentelė. 3 .Sakinio „Saulelė krypo vakarop“ daugiareikšmė morfologinė notacija ir *Gold standard* notacija.

<s>
Saulelė
dktv <saulelė>
  dktv mot.gim vnsk V
krypo
vksm <krypti(-sta,-o)>
  vksm teig nesngr tiesiog.nuos būt.kart.l vnsk IIIasm
  vksm teig nesngr tiesiog.nuos būt.kart.l dgsk IIIasm[6]
vakarop
prvks <vakarop>
  prvks
.
</s>
	<s>
	Saulelė
	<sub>
	dktv <saulelė>
	  dktv mot.gim vnsk V
	krypo
	<pre>
	vksm <krypti(-sta,-o)>
	  vksm teig nesngr tiesiog.nuos būt.kart.l vnsk IIIasm
	vakarop
	<mod>
	prvks <vakarop>
	  prvks
	.
	</s>

Kitas *Gold standard* įvertinimo etapas buvo tų pačių sakinių automatinis analizavimas sukurta priemone. Ir išėjimo duomenų palyginimas su *Gold standard* notacija. Duomenų palyginimas pateikiamas 4 lentelėje.

Lentelė. 4 Išėjimo duomenų palyginimas su *Gold standard* notacija

<s>
Saulelė
<sub>
dktv <saulelė>
dktv mot.gim vnsk V
krypo
<pre>
vksm <krypti(-sta,-o)>
vksm teig nesngr tiesiog.nuos būt.kart.1 vnsk IIIasm
vakarop
<mod>
prvks <vakarop>
prvks
.
</s>
	<s>
	Saulelė
	<sub>
	dktv <saulelė>
	dktv mot.gim vnsk V
	krypo
	<pre>
	vksm <krypti(-sta,-o)>
	vksm teig nesngr tiesiog.nuos būt.kart.1 vnsk IIIasm
	vakarop
	<atr>
	prvks <vakarop>
	prvks
	.
	</s>

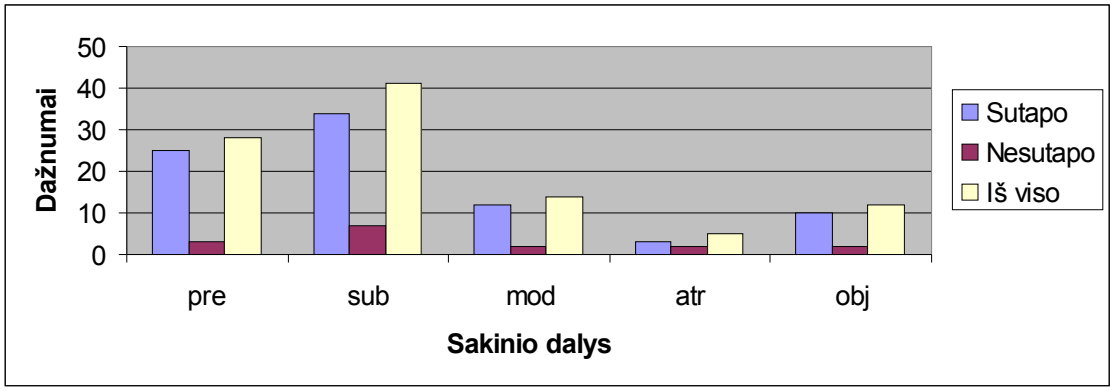
Iš 4 lentelės matome, kad analizės duomenys sutapo subjekto ir predikato nustatymo atvejai, o nesutapo aplinkybės nustatymo atveju (sistema priskyrė žodžiui „vakarop“ atributo reikšmę).

Tokiu būdu atlikus visų 20 sakinių analizę, gauti tokie rezultatai (žr. 5 lentelę):

Lentelė. 5. Tikslumo įverčio apskaičiavimo rezultatai.

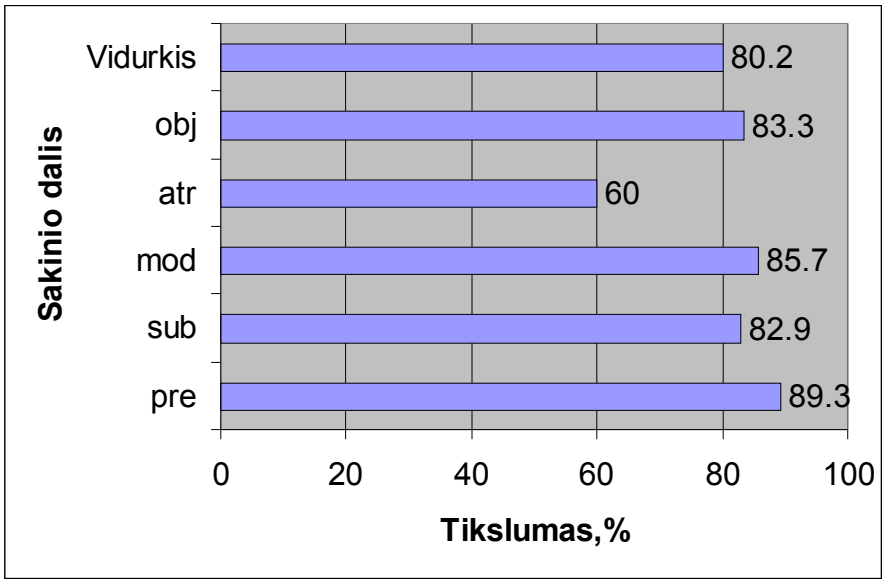
Sakinio dalis	pre	sub	mod	atr	obj
Sutapo	25	34	12	3	10
Nesutapo	3	7	2	2	2
Iš viso	28	41	14	5	12
Tikslumas, %	89,3	82,9	85,7	60	83,3

5 lentelės apibendrinti duomenys pateikti 22 pav. ir 23 pav.



[6] Ši morfologinė pažyma netinkama sakinio kontekstui ir sintaksinės analizės metu turėtų būti atmetama.

Pav. 22. Sakinių dalių sutapimo ir nesutapimo dažnumų pasiskirstymas.



Pav. 23. Sintaksinės analizės įrankio veikimo tikslumas.

Iš 22 pav. matome, kad nagrinėtuose sakiniuose didžiausia žodžių dalis tenka subjektui ir predikatui, o mažiausia – atributui. Predikato, objekto ir aplinkybės tinkamo pažymų nustatymo procentinis įvertis didžiausias, o atributo – mažiausias (žr. 23 pav.).

Akivaizdu, kad norėdami pakelti sistemos tikslumo rodiklį, turėtume susikoncentruoti į dvi žodžių klases: atributo – dėl mažiausio pažymos nustatymo procentinio įverčio ir subjekto – dėl didžiausio netinkamo pažymos priskyrimo dažnumo.

Sistemai analizuojant sakinius, sintaksinės pažymos nesutapimo atvejai galėjo atsirasti dėl šių priežasčių:

- Nepilno taisyklių rinkinio;
- Perteklinio taisyklių rinkinio;
- Algoritmo netobulumo.

Šių priežasčių žinojimas yra svarbus tolimesniam sistemos tobulinimui.

Sistemos veikimo tikslumo įvertis yra gana aukštas – 80,2 %. Pasaulyje tokių įrankių veikimo tikslumai įvairuoja nuo 67 % iki 97 % [4], [16].

4.4. Kompleksiškumo tyrimas

Sintaksiniame analizatoriuje paieška yra vienas svarbiausių uždavinių. Ji atliekama beveik visuose uždavinio sprendimo etapuose. Atliekant taisyklių ir apribojimų taikymą, tenka susidurti su keliais galimais sprendimo keliais. Paprasčiausia strategija tuo atveju yra pilnų perrinkimų vykdymas, tačiau tai pastebimai ilgina sintaksinės analizės trukmę.

Visiško perrinkimo paieškos algoritmo sudėtingumas deterministinei gramatikai yra $O(n^2)$ – paieška įtraukia n(n-1) žodžių poras. Žinoma, kad n artėjant į begalybę n(n-1) $\rightarrow n^2$ [27].

Tas pats pilno perrinkimo uždavinys žodžių junginių (bent pora žodžių) ir esamo standarto sintaksinių taisyklių (pvz., dvižodėms, trižodėms) suderinamumui įvertinti išauga iki $O(n^4)$ polinominio sudėtingumo.

Kitas svarbus sintaksinės analizės uždavinys yra daugiareikšmiškumo ribojimas. Natūraliausias būdas spręsti lokalius daugiareikšmiškumus yra grįžimo tuo pačiu keliu (angl. *backtracking*)[7] strategija [21] (jei reikalinga alternatyva, grįžti prie naujausios nenaudotos alternatyvos).

Pilnu perrinkimu grįsta paieška su *grįžimo tuo pačiu keliu* strategija, paremtos sintaksinės analizės algoritmo kompleksiškumas yra $O(n^3)$, taip yra todėl, kad gali reikti pakartoti visą O(n) procesą, kol bus išanalizuotas kiekvienas iš n žodžių.

Sistemos darbo laikui, o drauge ir uždavinio kompleksiškumui pagerinti, *sintaksinių taisyklių ir žodžių junginių atitikimo skaičiavimo etape*, buvo realizuotas sąrašų paieškos algoritmas, įvertinantis unikalumo ir projektiškumo apribojimus (angl. *list based search with uniquness and projectivity*) (pseudokodas pateikiamas 6 lentelėje).

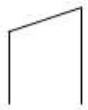
Lentelė 6. Sąrašų paieškos algoritmo pseudokodas.

[7] **backtracking** – tai strategija, skirta rasti nuo apibrėžto tinkamumo priklausantį sprendimą.

```
Duota analizuojamų žodžių sąrašas ir du pagalbiniai sąrašai: ValdSarasas and DarbSarasas:
(Inicializacija)
ValdSarasas := []; (žodžiai, kuriems nieko nevaldo)
DarbSarasas := []; (visi žodžiai)
Repeat
        (Paimamas žodis ir įdedamas į DarbSarasas )
        W := kitas analizuojamas žodis;
        DarbSarasas := W +DarbSarasas;

        (Ieškomi priklausomi nuo W žodžiai; Tikrinami vėliausiai idėti ValdSarasas elementai)

        for D := kiekvienam ValdSarasas elementui, pradedant nuo pirmojo
                begin
                        if D gali priklausyti W, tai
                                begin
                                        susieti D kaip priklausantį nuo W;
                                        ištrinti D iš ValdSarasas
                                end
                        else
                                nutraukti ciklą
                end;

                (ieškomi  W valdantys žodžiai)

                H := iš karto po W einantis žodis;
                loop
                        if W gali priklausyti H, tai
                                begin
                                        susieti W kaip priklausantį nuo H;
                                        nutraukti ciklą
                                end;
                        if H yra priklausomas, tai nutraukti ciklą;
                        H := valdantis H žodis
                end loop;
                if nebuvo rastas valdantis  W žodis, tai
                        ValdSarasas := W + ValdSarasas;
until neišanalizuojami visi žodžiai.
```

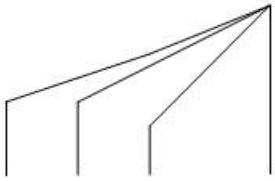
Šio sprendimo kompleksiškumo rezultatas nesikeis, taikant apribojimus ir pačiu blogiausiu atveju, kuomet reiks grįžinėti su kiekvienu žodžiu.

Pavyzdžiui, junginiui *mano ilgai lauktos atostogos* (16-18).



(16) mano ilgai lauktos atostogos

(17) mano ilgai lauktos atostogos



(18) mano ilgai lauktos atostogos

Paprastai kalboje retai aptinkami tokie atvejai, todėl uždavinio sprendimo kompleksiškumas ne visada bus lygus blogiausio atvejo kompleksiškumui.

Realizuotos sistemos kompleksiškumas yra $\leq O(n^3)$.

Realizuota *žodžių junginių* paieška su *grįžimo tuo pačiu keliu* strategija analizės algoritmo kompleksiškumą sumažino nuo $O(n^4)$ iki $O(n^3)$ (blogiausiu atveju).

5. IŠVADOS

1. Išanalizavus sintaksinės analizės sistemas, jų panaudojimo sritis, galimybes bei apribojimus buvo iškeltos sprendžiamo uždavinio problemos ir pasirinktas priklausomybių gramatikos būdas joms spręsti. Problemos, su kuriomis susiduriama sintaksinės analizės metu, yra morfologinis daugiareikšmiškumas ir sintaksinis daugiareikšmiškumas.
2. Pasirinktas sprendimo kelias suformuluotoms problemoms spręsti, remiasi priklausomybių gramatikos analizės, pagrįstos taisyklių taikymu, metodu. Pagrindiniai kriterijai, nulėmę tokį pasirinkimą, yra metodo tinkamumas tyrimo objektui (lietuvių kalbos analizei), sukurtos sistemos veikimo tikslumo gerinimo galimybė, pakartotinis panaudojamumas, sistemos išplečiamumas ir palaikomumas.
3. Suprojektuota ir realizuota sistema sėkmingai sujungia programų sistemų inžinerijos metodus ir kalbų technologijų srities žinias.
4. Realizacijoje buvo panaudotas greito nusileidimo lokalios paieškos algoritmas, jo pritaikymas – analizės *sakinio konstravimui* iš galimų sakinio žodžių junginių.
5. Sistemos darbo laikui ir uždavinio kompleksiškumui pagerinti sprendžiant žodžių junginių, atitinkančių taisykles, generavimo uždavinį, buvo realizuotas sąrašų paieškos algoritmas, įvertinantis unikalumo ir projektiškumo apribojimus. Realizuota *žodžių junginių* paieška su *grįžimo tuo pačiu keliu* strategija analizės algoritmo kompleksiškumą sumažino nuo $O(n^4)$ iki $O(n^3)$ (blogiausiu atveju).
6. Tyrimo metu sistema buvo įvertinta panašių sistemų atžvilgiu. Pastebėta, kad lietuvių kalbos sintaksinės analizės sistemos turi nemažai panašumų. Esminiai skirtumai, išskiriantys realizuotą sistemą iš kitų, yra sintaksinių taisyklių rinkinio panaudojimas, veikimo tikslumas, vertinant atskirų sakinio dalių atpažinimo vidurkį ir pats veikimo principas (algoritmas).

7. Eksperimento metu *Gold standard* metodu buvo įvertintas sistemos veikimo tikslumas. Buvo vertinamas atskirų sakinio žodžiu sintaksinių pažymų nustatymo tikslumas. Pastebėta, kad pažymų nustatymo klaidos dažniausios atributo (40 % atvejų) ir subjekto (17 % atvejų) žodžių grupėms, o rečiausios – predikatui (11 %). Sistemai analizuojant sakinius, sintaksinės pažymos nesutapimo atvejai galėjo atsirasti dėl tokių priežasčių: nepilno taisyklių rinkinio, perteklinio taisyklių rinkinio, algoritmo netobulumo.
8. Sukurtos sintaksinės analizės sistemos *veikimo tikslumo įvertis* yra 80,2 %. Kitoms kalboms sukurtos panašios sistemos veikia nuo 67 % iki 97 % tikslumu, todėl gautas rezultatas yra gana aukštas.
9. Tyrimai, susiję su sukurta sistema ir jos taikymu, pristatyti 2005 ir 2006 metų technologijų ir kalbų technologijų konferencijose.

Tolimesni darbai

Sukurtą eksperimentinę sintaksinės analizės sistemą planuojama tobulinti. Numatyti šie darbai:

- **Gerinti sistemos veikimo tikslumą** atskirų sakinio dalių analizės atžvilgiu ir viso sakinio analizės atžvilgiu. Sistemos veikimo tikslumo parametro gerinimas susijęs su šiais uždaviniais:
  - Sintaksės taisyklių rinkinio validavimu (lingvistinių resursų paruošimas);
  - **Sintaksės analizės algoritmo tobulinimu**;
- **Išplėsti sistemos funkcionalumą** – sukurti taisyklių įvedimo, redagavimo ir peržiūros sąsaja (sąsajos maketas pateiktas 1 priede);
- **Sukurti grafinį** sintaksės medžių **vaizdavimo modulį**;
- **Patobulinti** sukurtos *tikimos funkcijos* **įverčio skaičiavimo algoritmą**;
- **Praplėsti *medžio atrinkimo* modulį** taip, kad sistema grąžintų vieną sintaksinės analizės rezultatą;
- **Išplėsi sistemą ir pritaikyti ją medžių bankams** (angl. *tree banks*) **sudaryti**: išanalizuoti sakiniai gali būti peržiūrimi ir redaguojami, saugomi sintaksės medžių duomenų bazėje (medžių banke).

LITERATŪRA

[1] Ambrazas, V. (red.); K. Garšva; A. Girdenis (1996). *Dabartinės lietuvių kalbos gramatika*. 2-asis patais. leid. Mokslo ir enciklopedijų leidykla, Vilnius;

[2] Collins M. (2005). *Discriminative Reranking for Natural Language Parsing*., Computational Linguistics 31(1):25-69.

[3] Daudaravicius, V.(2002) *Lietuvių kalbos sakinio sintaksinė analizė*, Magistro tezės, VDU.

[4] Eisner, M. (1996). *Three new probabilistic models for Dependancy parsing*: an exploration, proceeding of COLLING-96, Copenhagen, p. 340-345.

[5] Engel, Ulrich. (1996). Tesnière mißverstanden. In Gertrud Gréciano and Helmut Schumacher, editors, *Lucien Tesnière - Syntaxe structurale et opérations mentales*, volume 348 of *Linguistische Arbeiten*. Max Niemeyer Verlag, Tübingen, p. 53-61.

[6] *Functional Dependency Grammar (FDG)* – Anglų kalbos sintaksinis analizatorius [žiūrėta 2005 10 30], interneto prieiga <http://www.ling.helsinki.fi/~tapanain/dg/eng/demo.html>

[7] Language and *Linguistics* (1998), red. R. E. Asher, vol. 2, 867– 872. Oxford: Pergamon Press, P.112-145.

[8] Grigonytė G., Rimkutė E. *Formal Specifications for a Dependency Grammar of the Lithuanian Language* – The Second Baltic Conference on Human Language Technologies proceedings. Tallinn, 2005, P. 237–242.

[9] Grigonytė G., Rimkutė E. *Automatinis lietuvių kalbos veiksmažodžių grupių atpažinimas* – konferencijos Informacinės technologijos 2005 pranešimų medžiaga, 2005, P. 315–320.

[10]Grigonytė G., Rimkutė E. *Priklausomybių gramatika pagrįstų lietuvių kalbos sintaksinių taisyklių išgavimas iš Dabartinės lietuvių kalbos tekstyno*. – X-osios tarpuniversitetinės magistrantų ir doktorantų konferencijos Informacinės technologijos pranešimų medžiaga, 2005, P. 65–67.

[11]Hays, David G. (1964). *Dependency theory: A formalism and some observations.* Language, 40:511-525.

[12]Hauser. R. (2001), *Foundation of Computational Linguistics*. Springer press. P.33-49, 125-139, 301-318.

[13]Hellwig, P., *Natural language parsers. A course in cooking*., [žiūrėta 2006 02 03], interneto prieiga <http://www.cl.uni-heidelberg.de/~hellwig/pars03.pdf>

[14]Jackendoff, Ray (1977) *X Syntax*. Cambridge, Mass.: MIT Press, P 12-15.

[15]Japenga, R. *Principles of software Driven User interface Desing for Business and Industrials Applications*. [žiūrėta 2006-03-15] prieiga internete: <http://www.microtoolsinc.com/articles.php>

[16]Järvinen, Timo and Tapanainen, Pasi. (1998). *Dependancy parser demo. University of Helsinki, Department of General Linguistics*, p. 1-2.

[17]Jurafsky D. and Martin J.(2000) *Speech and Language Processing*. Prentice Hall. P.285-499.

[18]*Lietuvių kalbos tekstynas*, [žiūrėta 2006-05-10] prieiga internete: <http://donelaitis.vdu.lt>.

[19]Mates, Benson (1961) *Stoic Logic*. Berkeley: University of California Press, p.68-82.

[20]Miyao Y., and Tsujii J.. (2005). *Probabilistic Disambiguation Models for Wide-Coverage HPSG Parsing*. In Proceedings of ACL-2005, pp. 83-90.

[21]Mitkov R.(2004). *Computational Linguistics*, Oxford press. P.25-91,178-249.

[22]Rimkutė E. *Morfologinio daugiareikšmiškumo tipologija*. – *Lituanistica*, 2003, Nr. 4 (56), P. 60–78.

[23]Robinson, Jane J. (1970). *Dependency structures and transformational rules*. *Language*, 46:259-285.

[24]Šveikauskienė D. *Formal Description of the Syntax of the Lithuanian Language*. Kaunas, Technologija*,* 2005, Vol. 34, No. 3, 245 – 256.

[25]Tapanainen, Pasi and Timo Järvinen. (1997). *A non-projective dependency parser*. In Proceedings of the 5th Conference on Applied Natural Language Processing, Washington, D.C., April. Association for Computational Linguistics.

[26]Valeckienė, A. (1998). *Funkcinė lietuvių kalbos gramatika*. Mokslo ir enciklopedijų leidybos institutas, Vilnius.

[27]Covington M. 2001 „*A fundamental algorithm for dependency parsing*“. Univerisity of Georgia.

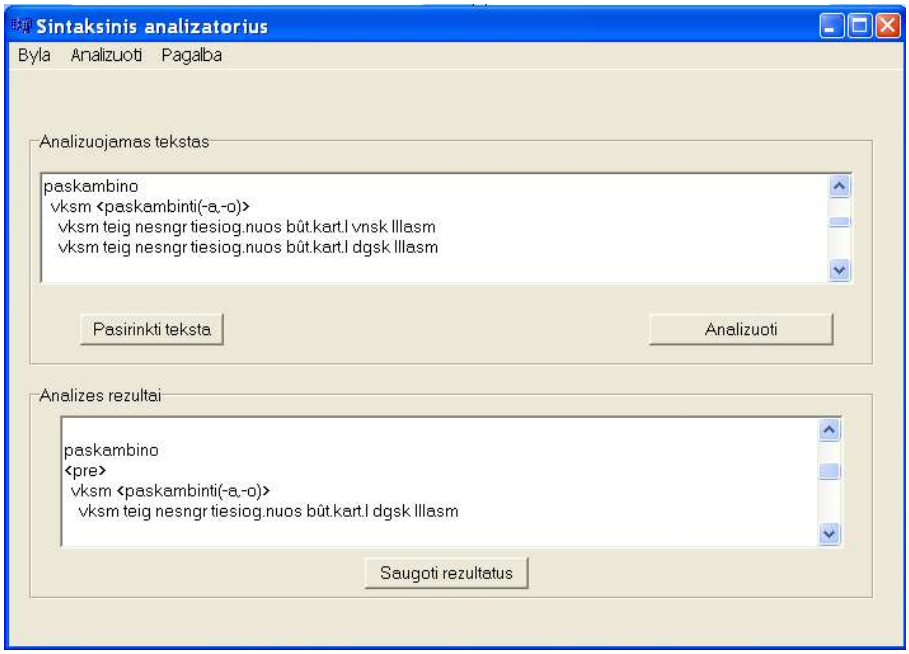
6. TERMINŲ IR SANTRUMPŲ ŽODYNAS

Terminas	Apibūdinimas
DLKT	Dabartinis lietuvių kalbos tekstynas.
Gold Standard	Sintaksinių analizatorių įvertinimo metodika.
Morfologinė notacija	Detali informacija apie žodį, pavyzdžiui: <w l="eiti(eina,ėjo)" m="finite verb, infinitive" l="eiti(eina,ėjo)" m="finite verb, participle>**eiti**<w>
PG	Formalioji sintaksės gramatika (Dependency Grammar).
Priklausomybių medis	Logiška, vizuali sintaksinė sakinio struktūra, paremta priklausomybių gramatika.
Projektiškumas	Priklausomybių medžio savybė nusakanti sąsajų nesikirtimą sintaksinėse priklausomybių struktūrose.
RUP	Rational Software pristatyta programinės įrangos kūrimo metodika (Rational Unified Process).
Sintaksė notacija	Detali informacija apie sakinį arba žodžių junginį (sakinio dalys, priklausomybės, papildomi parametrai).
Tekstynas	Elektroninių tekstų rinkinys
UML	Unifikuota modeliavimo kalba (Unified Modeling Language).

7. PRIEDAI

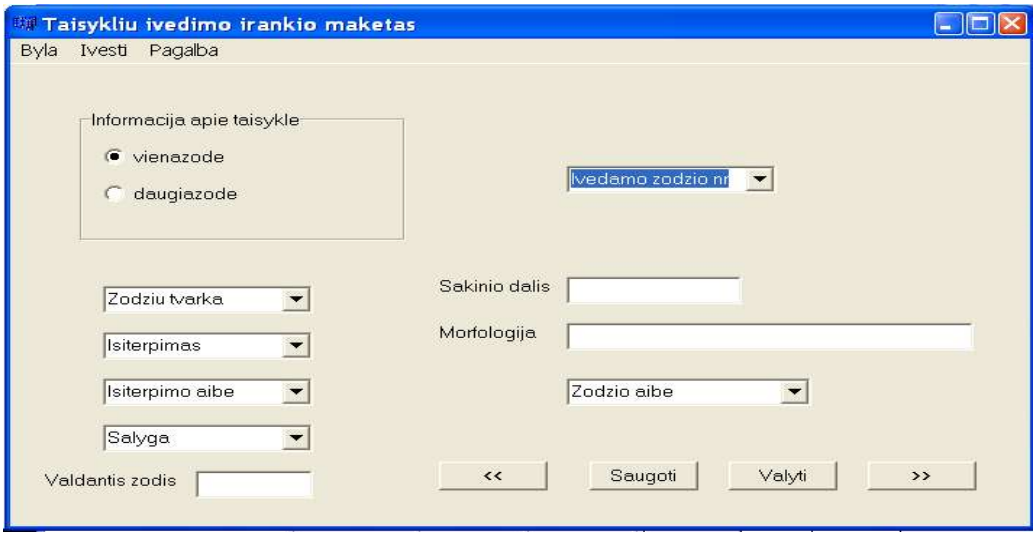
7.1. Priedas.1. Vartotojo sąsajos langai

Sintaksinio analizatoriaus demonstracinės versijos langas pateikiamas pav.24.



Pav. 24. Demonstracinės versijos langas.

Planuojamo įgyvendinti taisyklių įvedimo ir peržiūrėjimo lango maketas pateikiamas pav.25.



Pav. 25 Taisyklių įvedimo ir peržiūrėjimo lango maketas

7.2. 2 Priedas. Sistemos naudojamų duomenų pavyzdžiai

7.2.1. Analizuojamas tekstas

```
Karo
  dktv <karas>
    dktv vyr.gim vnsk K
  vksm <karoti(-o,-ojo)>
    vksm teig nesngr tiesiog.nuos esam.l vnsk IIIasm
    vksm teig nesngr tiesiog.nuos esam.l dgsk IIIasm
  vksm <karti(-ąra,-aro)>
    vksm teig nesngr tiesiog.nuos būt.kart.l vnsk IIIasm
    vksm teig nesngr tiesiog.nuos būt.kart.l dgsk IIIasm

audra
  dktv <audra>
    dktv mot.gim vnsk V
    dktv mot.gim vnsk Įn

šioje
  Įvrd <šis>
    Įvrd neįvardž mot.gim vnsk Vt

vietovėje
  dktv <vietovė>
    dktv mot.gim vnsk Vt

buvo
  vksm <būti(yra,buvo)>
    vksm teig nesngr tiesiog.nuos būt.kart.l vnsk IIIasm
    vksm teig nesngr tiesiog.nuos būt.kart.l dgsk IIIasm

itin
  prvks <itin>
    prvks

žiauri
  bdvr <žiaurus>
    bdvr teig nelygin.l neįvardž mot.gim vnsk V
.

Vieną
  dktv <viena>
    dktv mot.gim vnsk G
  tikr dktv <Viena>
    tikr dktv mot.gim vnsk G
  bdvr <vienas>
    bdvr teig nelygin.l neįvardž vyr.gim vnsk G
    bdvr teig nelygin.l neįvardž mot.gim vnsk G
  sktv <vienas>
    sktv kiekin vyr.gim vnsk G
    sktv kiekin mot.gim vnsk G
  įvrd <vienas>
    įvrd vyr.gim vnsk G
    įvrd mot.gim vnsk G

jų
  įvrd <jis>
    įvrd neįvardž vyr.gim dgsk K
    įvrd neįvardž mot.gim dgsk K

pažinojau
  vksm <pažinoti(-o,-ojo)>
    vksm teig nesngr tiesiog.nuos būt.kart.l vnsk Iasm
  vksm <pažinoti(-ja,-jo)>
    vksm teig nesngr tiesiog.nuos būt.kart.l vnsk Iasm
.
Jie
  įvrd <jis>
    įvrd neįvardž vyr.gim dgsk V

kažkur
  prvks <kažkur>
    prvks

paskambino
  vksm <paskambinti(-a,-o)>
    vksm teig nesngr tiesiog.nuos būt.kart.l vnsk IIIasm
    vksm teig nesngr tiesiog.nuos būt.kart.l dgsk IIIasm

ir
  prvks <ir>
    prvks
  dll <ir>
    dll
  jngt <ir>
    jngt

tarėsi
  vksm <tartis(-iasi,-ėsi)>
    vksm teig sngr tiesiog.nuos būt.kart.l vnsk IIIasm
    vksm teig sngr tiesiog.nuos būt.kart.l dgsk IIIasm
.

Priežasties
  dktv <priežastis>
    dktv mot.gim vnsk K

nepasakė
  vksm <nepasakyti(-o,-ė)>
    vksm neig nesngr tiesiog.nuos būt.kart.l vnsk IIIasm
    vksm neig nesngr tiesiog.nuos būt.kart.l dgsk IIIasm

ir
  prvks <ir>
    prvks
  dll <ir>
    dll
  jngt <ir>
    jngt

vėliau
  prvks <vėliai>
    prvks aukštesn.l
  prvks <vėlai>
    prvks aukštesn.l
  vksm <velti(-elia,-ėlė)>
    vksm teig nesngr tiesiog.nuos būt.kart.l vnsk Iasm
.
```

7.2.2. Naudojamos taisyklės

(s;s1;1)
(s;[dktv,V|tikr.dktv,V];[-1])
(*)
(*)

(s;s2;1)
(s;[įvrd,V];[įvrd_dktv])
(*)
(*)

(s;s3;1)
(s;[sktv,V];[sktv_kiekin_1-9|sktv_kiekin_nuo_11|
sktv_kuopin])
(*)
(*)

(s;s4;1)
(s;[dktv,dgsk,K];[-1])
(*)
(*)

(s;s5;1)
(s;[dktv,N|tikr.dktv,N];[-1])
(*)
(*)

(s;s6;1)
(s;[bdvr,N];[-1])
(*)
(*)

(p;p7;1)
(p;[bndr];[-1])
(*)
(*)

(p;p8;1)
(~s13)
(*)
(*)

(p;p9;1)
(p;[vksm];[-1])
(*)
(*)

(a;a1;1)
(a;[bdvr|dlv|įvrd|sktv];[-1])
(*)
(*)

(a;a2;1)
(a;[dktv,K|tikr.dktv,K];[-1])
(*)
(*)

(a;a3;1)
(a;[dktv,N|tikr.dktv,N];[-1])
(*)
(*)

(o;o1;1)
(o;[dktv,K|dlv,įvardž,K|bdvr,įvardž,K|sktv,K|įvrd,K];
[įvrd_dktv])
(*)
(*)

(o;o2;1)
(o;[dktv,N|dlv,įvardž,N|bdvr,įvardž,N|sktv,N|įvrd,N];
[įvrd_dktv])
(*)
(*)

(o;o3;1)
(o;[dktv,G|dlv,įvardž,G|bdvr,įvardž,G|sktv,G|įvrd,G];
[įvrd_dktv])
(*)
(*)

(o;o4;1)
(o;[dktv,Įn|dlv,įvardž,Įn|bdvr,įvardž,Įn|sktv,Įn|įvrd,Įn];
[įvrd_dktv])
(*)
(*)

(o;o5;1)
(~p7)
(*)
(*)

(p;ppp1;2)
(p;[vksm];[vksm_modal])
(~pp1)
(2;1;[-1];[-1];p)

(p;pp2;2)
(p;[vksm];[vksm_modal])
(~p7)
(1;1;[-1];[-1];p)

(p;pp3;2)
(p;[vksm,IIIasm];[vksm_beasmen_bndr])
(~p7)
(1;1;[-1];[-1];p)

(a;aa14;2)
(a;[prvks,aukšč.l];[prvks_kiekyb_kokyb_pagrind])
(a;[bdvr|dlv];[-1])
(2;1;[-1];[-1];a)

(a;aa15;2)
(a;[prln];[prln_iš])
(a;[dktv,K];[-1])
(3;1;[-1];[-1];a)

7.3. Autorės publikacijos nagrinėjama tema